One-dimensional spline smoothing in insurance

By

Eirik Smidt Sagstuen

THESIS

For the degree of

MASTER OF SCIENCE

(Master's degree in Modelling and Data Analysis)

Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

April 2014

Abstract

There are lots of special techniques and distribution models used to solve different problems in the insurance industry today. Learning the theory behind all of them is time consuming and leaves little time to analyze results. There are also few techniques and models which work well in situations with little data, and just using the empirical distribution function can lead to the underestimation of future liabilities. This thesis deals with spline smoothing models and their possible applications in the insurance industry. Spline models are simply put piecewise defined polynomial functions with smooth derivatives. When using spline models no "view" is put on the data, which can be an advantage in situations with little and/or long tailed data.

The main objectives of the thesis are;
1. To highlight that there is a need for a general technique which can make models designed for specific purposes obsolete.
2. To show that spline models used together with link-functions can be such a general technique.
3. Write compact and easy to understand programmes that can easily be implemented into standard software.

Acknowledgements

This thesis concludes my master's degree in science. Behind every achievement there's more than one person, and this one is no exception. I'm deeply indebted to my parents for being passionate supportive of me pursuing an advanced degree. My employer Codan Forsikring and especially my manager Mark Welsh has also been of great support. Although having me on as a full-time staff, Codan and Mark have given me the opportunity to work on my masters degree during office hours, and been flexible on deadlines. I'd also like to thank my supervisor Erik Bølviken for all the help he gave during my work with the thesis.

Table of Contents

Chapter 1

Introduction

Parametric models play a huge role in the insurance industry. The Gomperz-Makeham model is widely used to model mortality probabilities in life insurance, and the Pareto model is an example of a parametric model that's used to model claim sizes in general insurance. Parametric models smooth and provide estimates with lower uncertainty than the non-parametric ones. The backbone is however that a "view" is put on the data that is often not justifiable, this introduces an element of systematic error which ultimately can result in too high or too low insurance reserves and premiums. Another drawback is that some of the parametric models contain too many parameters, depending on the amount of data; there could be much uncertainty in the parameter estimates as well. Examples of models with excessive number of parameters are a five parameter model of mortality intensity by W.F Perks introduced in 1932 and an eight parameter model of mortality odds by Heligman and Pollard introduced in 1980, see Pitacco(2004) for more information about these two models. When dealing with small sample sizes like Norwegian life insurance portfolios, the parameter estimates in models like the two just mentioned might be uncertain. I will in this thesis work with spline models which smoothes the data without putting any distribution on it. Spline models are simply put piecewise defined polynomial functions with smooth derivatives and potenially few parameters. The aim of using spline models is to establish a general technique for modelling of different phenomena; a spline model that together with transformations and constant terms adapted to the modelling situation will make parametric models designed for specific purposes obsolete. Having a well functioning general technique could potentially save actuarial students and others a lot of study-time, study-time that might be better spent trying to analyze the results from a model rather than spending time trying to understand the theory behind it. Spline models can easily be implemented into automatic processes which are highly valuable for modellers and they're crucial for automatic systems where modelling is taken care of by a computer. In a world were modellers are a scarce resource, having an automatic procedure for fitting models to different portfolios is highly valuable. This thesis focuses

highly on the practical side of spine models, i.e how spline models fit to datasets, empirical and simulated. Since insurance data usually are long tailed, all datasets used in this thesis are as well and the emphasis has been to look at the fit of the spline models to the tail of the data. During the work with the thesis a lot of time went to writing compact and transparent code in R. A sub goal of the thesis was to write the code as easy to use as possible so that it could be implemented into a package in R. All codes and datasets used to produce the results in this thesis are in the appendix, and the reader is encouraged to take a look at it. The thesis has one chapter on the applicability of spline models in life insurance and two chapters about general insurance. While the chapter on life insurance only focuses on goodness-of-fit, the two chapters on general insurance deals with applications to insurance such as reserves as well.

Chapter 2

Spline models

2.1 Introduction

The word spline has its origin from the ship building industry where a spline was a long, thin and flexible strip of wood or other material used to design the smooth curvatures of the ship's hull. The mathematical invention of spline models is credited to the Romanian-American mathematician Isaac Jacob Schoenberg who published a research article on the topic in 1946. A spline model can be defined as a piecewise polynomial function which possesses a high degree of smoothness at the places where the polynomial pieces connect, these places are called knots, and together these knots form a grid. Spline models have applications to multiple dimensions, but I will in this thesis only work with one-dimensional spline models and their applications to insurance data

2.2 Definition

A spline model F defined on the interval $[xc_1{:}xc_N]$ with subintervals $xc_1 < xc_2 < .. < xc_{N-1} < xc_N$ is a piecewise polynomial function:

$$F(x) = \begin{cases} F_1(x) \text{ if } xc_1 \le x \le xc_2 \\ F_2(x) \text{ if } xc_2 \le x \le xc_3 \\ \vdots \\ F_N(x) \text{ if } xc_N \le x \le \infty \end{cases}$$

$xc_1 < xc_2 < .. < xc_{N-1} < xc_N$ are called knots, and together they form a grid.

For a cubic spline every function $F_j(x)$ $j = 1, \dots, N$ is a third order polynomial:

$F_j(x) = a_j(x - xc_j)^3 + b_j(x - xc_j)^2 + c_j(x - xc_j) + d_j$ for $j = 1, \dots, N$

This gives 4N parameters to estimate, if a cubic spline is twice continuously differentiable, it has the three properties:

$F_{j-1}(xc_j) = F_j(xc_j)$, $F'_{j-1}(xc_j) = F'_j(xc_j)$ and $F''_{j-1}(xc_j) = F''_j(xc_j)$ for $j = 2, \dots, N$

Based on these three properties we can form 3(N-1) equations, provided they're linearly independent the effective number of parameters to be estimated can be reduced to 4N - 3(N-1) = N+3.

2.3 B-splines

The cubic spline model from chapter 2.2 can be rewritten as:

$$F(x) = d_1 + c_1x + b_1x^2 + a_1x^3 + \sum_{j=2}^{N} a_j(x - xc_j)_+^3 \quad (2.1)$$

Setting $b_1 = c_1 = d_1 = 0$ we arrive at the spline model which will be the basis for spline models used in this thesis:

$$F(x) = \sum_{j=1}^{N} a_j(x - xc_j)_+^3 \quad (2.2)$$

This spline model is a lot easier to implement into optimization schemes than the spline model introduced in chapter 2.2 and is called a basic spline(b-spline).

2.4 Fitting spline models to data

All sensible criteria's can be used in order fit the spline models to data. In this thesis I used the maximum likelihood method to fit spline models to mortality data and the least squares method to fit spline models to claim size data. The two methods were chosen because they're the most widely used in practise. When using the maximum likelihood method I found that the optimization procedure proved to be much less sensitive to bad start values and converged faster when partial derivatives of the objective function were provided.

2.5 Optimization methods.

The programming language R was used when working with this thesis. I used the optim function to fit the spline functions to the data-sets. The optim function performs minimisation by using quasi-newton methods. In short, quasi-Newton methods are algorithms aiming to find the stationary point of a function. Unlike Newtons method, quasi-Newton methods estimate the inverse of the hessian matrix directly, which is more effective when maximizing a function with several variables.

2.6 Grid-structure

The grid-structures, that is the values of $xc_1 < xc_2 < .. < xc_{N-1} < xc_N$ used when fitting spline models to data in this thesis were all more or less found by trial-and-error. To begin with equidistant knots were chosen, but choosing those grids didn't give as good a fit to datasets as when non-equidistant knots were chosen. The biggest reason for this is the datasets used in this thesis, they're all long-tailed. The grid-structure must therefore reflect that, and be more centred around the tail of the data. The specific strategy used for choosing knots in this thesis was to plot the data and look for "breaking points" in the data e.g where the higher values start and place a knot there and another where the extreme values start. Grid-structures were different for quadratic and cubic splines models. This was because it's easier to catch tails with cubic splines and therefore the knot that was going to catch the higher values could be placed on a higher value for a cubic then a quadratic spline model.

Chapter 3

Mortality modelling

3.1 Introduction

One of the biggest problems on the liability side of pension insurance is finding out how long customers live. Since there are big differences in average mortality probabilities between the sexes, occupations and so forth, country-averages cannot be used. In other words; homogenous data are usually hard to come by (unless you've been insuring the same union for 50 years). Suppose there are n_x individuals in age x with y_x dying during a given year. The basic estimate of the mortality probability for age x is then $q_x = y_x/n_x$. Such basic estimates can be uncertain when n_x or y_x is small, which is the case in many pension portfolios. This chapter will introduce a spline model for the modelling of one-year mortality probabilities and show results from fitting a quadratic and cubic spline model to a simulated dataset of mortality probabilities. The Gompertz-Makeham model will also be featured, a model that is frequently used for modelling mortality probabilities in the insurance industry. The three models will be fitted to the same dataset and compared on the basis of goodness-of-fit tests, which in this case is the Aikaikes information criterion and graph analysis.

3.2 Spline model for mortality probabilities

A spline model of degree k for modelling mortality probabilities can be:

$$q_x = \frac{e^{S(x)}}{1+e^{S(x)}} = \frac{1}{1+e^{-S(x)}} \text{ where } S(x) = d_1 + \sum_{j=1}^{N} a_j \left(xc_j - x\right)_+^k \qquad (3.1)$$

In order to secure that q_x has a value between 0 and 1 for all x a transformation must be used, in this case the logit-function $S(x) = \log\left(\frac{q_x}{1-q_x}\right)$ was used. This is just one of many functions which can be used as transformations for a spline model and e.g linear predictors, they're often called link-functions. Another link-function is the log-function; $S(x) = \log(q_x)$, see Nelder and McCullagh (1989) for more examples of link-functions. The spline itself $S(x)$ is a constant term d_1 plus the representation introduced in chapter 2.3. The mortality probabilities can be found by optimizing numerically with respect to $d_1, a_1, \dots, a_N$ through maximizing the log-likelihood function:

$$L = \sum_x (y_x \log(q_x) + (n_x - y_x)\log(1 - q_x)) \qquad (3.2)$$

The sum in (3.2) is over all x for which there are data; i.e. for which $n_x > 0$. In the results from this procedure shown later in this chapter $N = 3$ non-equidistant knots were chosen. This grid-structure was determined by trial-and-error and was different for quadratic and cubic spline models. The reason for adding d_1 to the spline representation is to stabilize the maximization for ages with low exposure; some ages might not even have any deaths in a given year. This is usually the case in insurance schemes for younger ages. The optimization procedure proved to work better when $a_1, \dots, a_N$ were entered through the link-function: $\tan^{-1}(a_j)\,(2A/\pi)$ for $j = 1, \dots, N$ where A is a maximum specified for $|a_j|$. By setting A=0.1 in the optimization procedure, the constraint didn't seem to affect the estimation of a_j for quadratic and cubic splines. With the second link-function incorporated S(x) will now be:

$$S(x) = d_1 + \sum_{j=1}^{N} \tan^{-1}(a_j) \left(\frac{2A}{\pi}\right) (xc_j - x)_+^k$$

and by rearranging the log-likelihood can be simplified to:

$$L = \sum_x \left(y_x S(x) - n_x \log\left(1 + e^{S(x)}\right)\right) \qquad (3.3)$$

The partial derivatives of the log-likelihood function are used in the maximization procedure in order for the procedure to work even though bad start values are chosen. The partial derivatives of the log-likelihood function (2.3) are the following:

$$\frac{dL}{dd_1} = \sum_x \left(y_x - n_x \frac{e^{S(x)}}{1 + e^{S(x)}}\right)$$

$$\frac{dL}{da_j} = \sum_x \left(y_x \frac{1}{\left(1+a_j^2\right)} \frac{2A}{\pi} (x - xc_j)_+^2 - n_x \frac{1}{\left(1+a_j^2\right)} \frac{2A}{\pi} (x - xc_j)_+^2 \frac{e^{S(x)}}{1+e^{S(x)}}\right) \text{ for } j=1,..,N$$

3.3 Simulated datasets

In the next chapters results from fitting quadratic and cubic spline models and the Gompertz-Makeham model to mortality probabilities will be shown. The dataset used to fit the models are simulated and based on a real data-set with mortality data for Norwegian males in 2011. The dataset of the mortality data for Norwegian males contains 1.9 million observations of males aged 18-105 and the number of deaths incurred for each age in 2011. The age-distribution in the dataset for Norwegian males is illustrated left in figure 3.1. Such a distribution is not applicable to a pension portfolio, so an alternative age distribution is introduced $n_x = c * e^{-\gamma(x-50)}$ where c is determined so that $\sum_x n_x = J$, J is the size of the portfolio and n_x is the number of policyholders in age x. The number of deaths for age x in the portfolio is simulated by $y_x \sim \text{bin}(n_x, Y_x/N_x)$ where Y_x/N_x is the observed death-rate for age x in the Norwegian male data. J were chosen to be 100 000 when making the simulated dataset, to reflect a fairly common pension portfolio size.

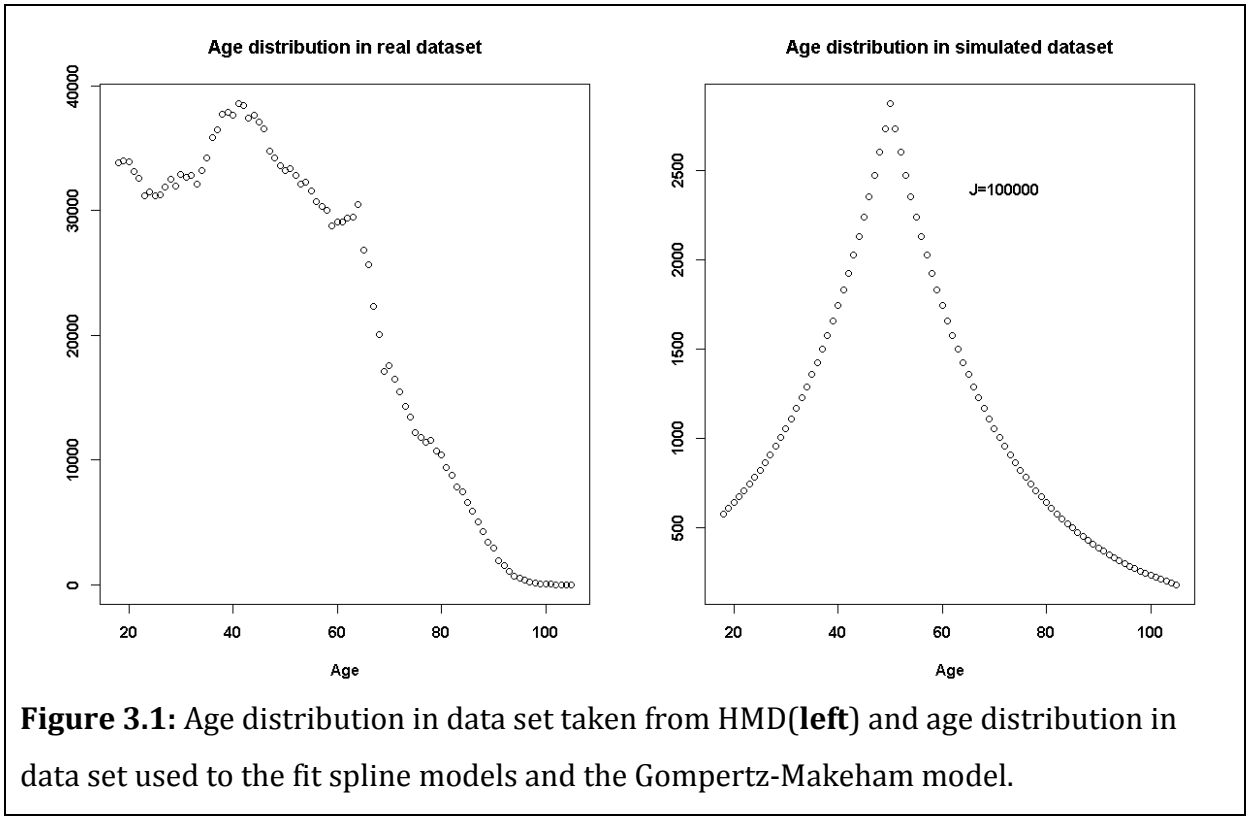

Figure 3.1: Age distribution in data set taken from HMD(**left**) and age distribution in data set used to the fit spline models and the Gompertz-Makeham model.

3.4 Quadratic spline model

By setting k=2 in the spline model defined as (3.1) we get a quadratic spline model:

$$q_x = \frac{e^{S(x)}}{1+e^{S(x)}} = \frac{1}{1+e^{-S(x)}} \text{ where } S(x) = d_1 + \sum_{j=1}^{N} a_j (xc_j - x)_+^2 \qquad (3.4)$$

This model was inserted into the maximization procedure defined as (3.3). The dataset used was the simulated dataset of mortality probabilities introduced in chapter 3.3. The maximization itself of the log-likelihood function was done in the statistical programming language R, which has been used to create every figure and graph in this thesis. See appendix for code and Steenbergen (2006) for tips on notation for coding of the maximum likelihood function in R and use of the optim function. Below are the results from fitting quadratic spline model to the simulated dataset.

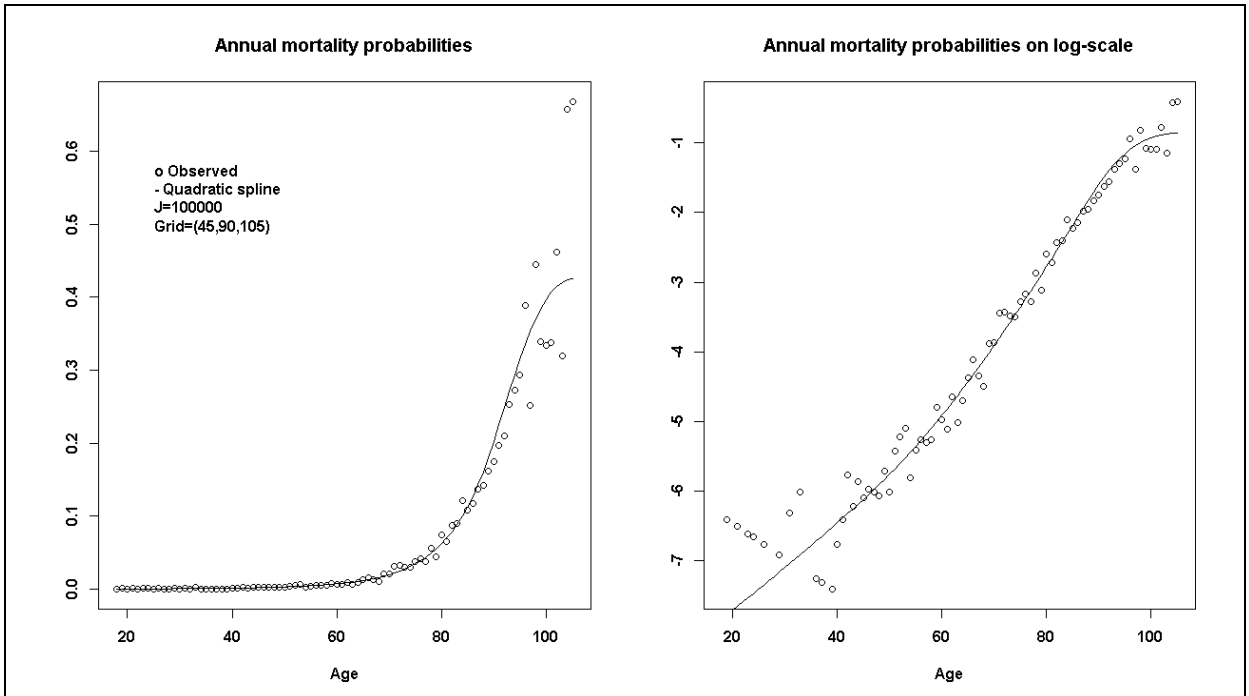

Figure 3.2: Quadratic spline model fitted to mortality data on original scale(**left**), and log-scale(**right**).

The plot of the annual mortality probabilities on log-scale is added to highlight the bad fit for lower ages. This has to do with very low exposure for these ages, and had it not been for the constant term in the spline model, the fit would have been even worse.
By looking at figure 3.2 it can be claimed at the four-parameter quadratic spline model fit the simulated mortality probabilities fairly well except for the absolute highest and lowest ages.

3.5 Cubic spline model

By setting k=3 in the spline model defined as (3.1) we get a cubic spline model:

$$q_x = \frac{e^{S(x)}}{1+e^{S(x)}} = \frac{1}{1+e^{-S(x)}} \text{ where } S(x) = d_1 + \sum_{j=1}^{N} a_j \left(xc_j - x\right)_+^3 \qquad (3.5)$$

The cubic spline model was fitted to the simulated mortality probabilities the same way as the quadratic one and the results are illustrated below.

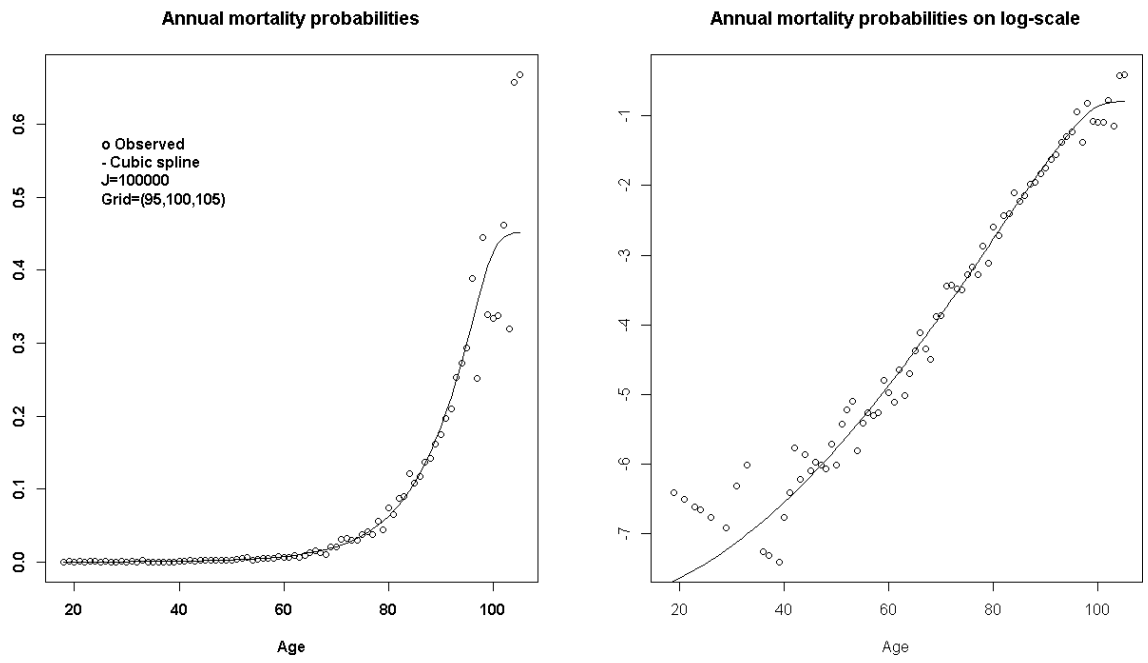

Figure 3.3: Cubic spline model fitted to mortality data on original scale(**left**) and log-scale(**right**).

The cubic spline model has more or less the same fit as the quadratic one, except that it fits the mortality probabilities for the older ages a little better. This however would mean nothing for a pension portfolio. So few people live up to those ages that the financial impact of choosing the quadratic spline model as a basis for premium calculations instead of the cubic one would be minimal. The most important from an actuarial viewpoint is that the models fit the mortality probabilities well for ages 40 – 90, which they both do.

3.6 Gompertz-Makeham model for mortality probabilities

The Gompertz-Makeham model is perhaps the most widely used model for mortality probabilities, even though its inception goes back to the 19th century. The Gompertz-Makeham model for a mortality probability in age x can be represented as:

$$q_x = \log(1 - \exp(-\theta_0 - \frac{\theta_1}{\theta_2}(e^{\theta_2} - 1)e^{\theta_2 x})) \qquad (3.6)$$

The probabilities q_x can be found by optimizing numerically with respect to θ_0,θ_1,θ_2 through maximizing the log-likelihood function defined as equation (3.3). The optimization procedure proved to work better when θ_0,θ_1,$\theta_2$were entered through the transformation e^{θ_i} for $i = 1,2,3$. Below are the results from fitting the Gompertz-Makeham model to the same simulated dataset as was used to fit the two spline models.

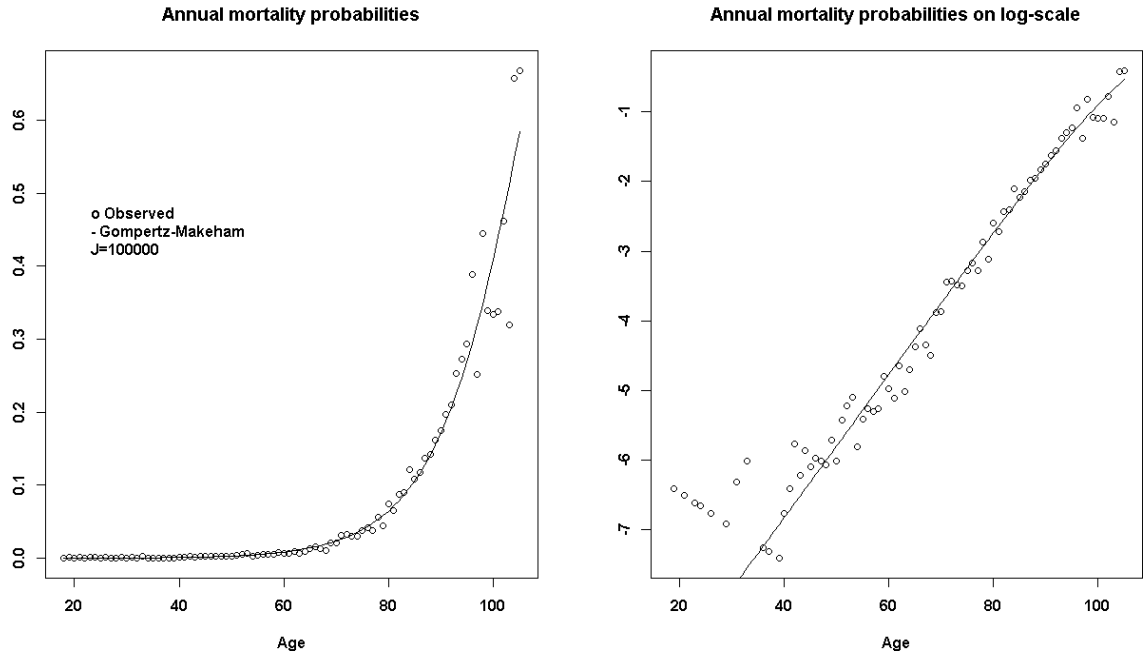

Figure 3.4: Gompertz-Makeham model fitted to mortality data on original scale(**left**) and log-scale(**right**).

The Gompertz-Makeham model fits the mortality probabilities for the highest ages far better than the two spline models. On the other hand, by looking at the graph of mortality probabilities on log-scale, one can see that the Gompertz-Makeham model does not fit the mortality probabilities for the lowest ages as well as the two spline models. Just from looking at the three figures, it's hard to determine which model is the best.

3.7 Comparison of the three models

Since it was hard to determine just by looking at the graphs in the earlier sections, we can use Akaikes information criterion (AIC) to determine which of the three models fits the mortality probabilities the best. AIC deals with the trade-off between the goodness-of-fit (maximization of the log-likelihood function) and the complexity of the model (number of parameters).

$$AIC = -2L + 2k$$

L is the maximised value of the log-likelihood function defined in equation (3.3) and k is the number of parameters in the model. For Akaikes information criterion(AIC) we get the following scores for the spline models and the Gompertz-Makeham model when J=100000:

Table 3.1: AIC values.
Quadratic spline model: $-2 \cdot (-7386) + 2 \cdot 4 = 14780$
Cubic spline model: $-2 \cdot (-7378) + 2 \cdot 4 = 14764$
Gompertz-Makeham model: $-2 \cdot (-7372) + 2 \cdot 3 = 14750$

The Gompertz-Makeham has the highest log-likelihood function value of the three models, this means that it has the best fit to the mortality probabilities just by looking at the value of the log-likelihood function. AIC introduces a penalty term for over fitting by taking into account how many parameters it is in the model. The spline models have one variable more than the Gompertz-Makeham. The AIC does not say anything about the quality of the model with the spline function in an absolute sense. But by looking at the graphs in the earlier sections of this chapter we can tell that all three models have a good fit to the data.

Chapter 4

Claims severity modelling I

4.1 Introduction

Severity is a word commonly used for the amount of a claim in general insurance. It's common to model the frequency and severity for claims separate, and one of the hardest phenomena to model is the severity for property insurance claims. Property insurance portfolios are usually very diverse, an example can be an insurance portfolio where 98% of the properties are regular houses with insurance limits of 1-2 million $, but the portfolio also contains a couple of mansions with limits up to 5-10 million $. Most of the claims in that portfolio will have a severity below 2 million $, but there might also be some which will be far greater. When setting aside insurance reserves for a portfolio like the one above it is important to have a model for the claims severity. But there are few models that fit well to data from such a portfolio. A common distribution model used to model property claims severity are the gamma distribution model. The problem with using a distribution model like the gamma one is that a view is put on the data that's often not justifiable and the fit of the model might not be great either. One can also just use the empirical distribution of the claims severity when calculating insurance reserves, but a problem with using the empirical distribution is that no future claim can have a bigger severity than the historical claims. In a portfolio that takes in bigger risks than it already has, this can lead to under reserving. This chapter will introduce spline models which can be used for modeling of claims severity. Spline models puts no view on the data, and by incorporating tail distribution functions, they might be able to fit well to even very long tailed datasets. This will be investigated by looking at the spline models ability to estimate the skewness in the underlying distribution. This chapter will introduce spline models with and without tail conditions for modeling of severity data.

4.2 Spline model for claims severity

Let $y_1, \dots, y_n$ be historical claims in a property insurance portfolio from which we want to determine a model for the underlying random variable Y. Let F(y) be the cumulative distribution function of Y and $F^{-1}(u)$ its percentile function. If $\hat{F}(y)$ and $\hat{F}^{-1}(u)$ are their estimates, a way to verify a model would be to compare the observations in ascending order $y_{(1)} \leq .. \leq y_{(n)}$ with $\hat{F}^{-1}(u_1), \dots, \hat{F}^{-1}(u_n)$, where $u_i = (i - 0.5)/n$. If they match, the fit can be deemed to be a good one. One way to make this process automatic is to introduce a cubic spline model:

$$\hat{F}^{-1}(u) = \sum_{j=1}^{N} a_j (u - xc_j)_+^3 + u * c_1 \qquad (4.1)$$

Where $xc_1 < \dots < xc_N < 1$ are knots which form a grid, with $xc_1 = 0$ so that $\hat{F}^{-1}(0) = 0$. The linear term $u * c_1$ is added to reduce the convexity of the model. Coefficients $a_1, \dots, a_N$ and c_1 are determined so that they minimize:

$$Q = \sum_{i=1}^{n} (\hat{F}^{-1}(u_i) - y_{(i)})^2 \qquad (4.2)$$

This optimization method is called least squares and is easily implemented into R. Below are results from fitting the spline model to simulated gamma and exponentially distributed data. These two distributions are used because they have properties that to some extent reflect the claims severity distribution of a property insurance portfolio.

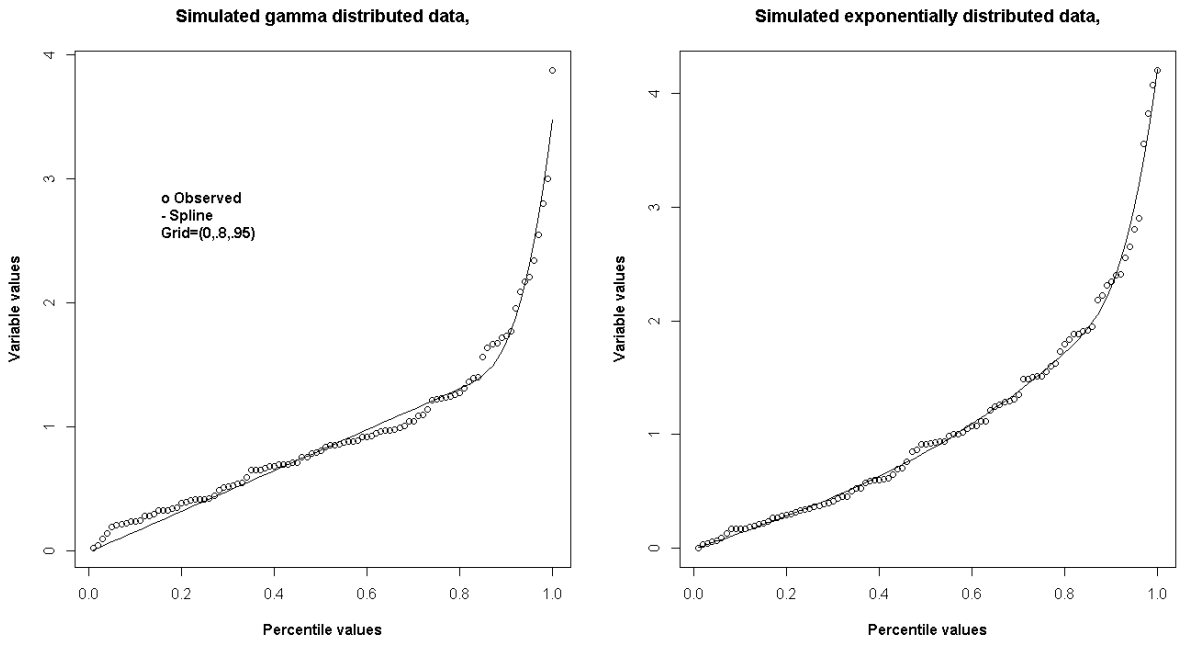

Figure 4.1: Spline model fitted to gamma-distributed data(**left**) and exponentially distributed data(**right**).

By looking at the two graphs in figure 4.1 we can conclude that the spline model fits the simulated data well, especially for exponentially distributed data. But the spline model isn't quite able to catch the right tail of the gamma distribution. This might be possible by implementing a tail distribution in the spline model, and the focus in the next section will be on this.

4.3 Incorporating tail distributions.

In chapter 4.2 a spline model was introduced as an approximation to the percentile function of the underlying data for which the distribution is often unknown, especially for small datasets. The results in figure 4.1 showed that the spline model didn't fit well to the tail on one of the datasets and it was proposed that an over the threshold distribution should be implemented. Implementing such a distribution is basically the same as fitting the data over a certain value b to a specified distribution. We can use Pickands teorem to illustrate the theory, if y>b then:

$$\mathrm{P}(\mathrm{Y} > y) = \mathrm{P}\big(\mathrm{Y} > y \,\big|\, Y > b\big)\mathrm{P}(\mathrm{Y} > b) = \Big(1 - \hat{\mathrm{F}}(\mathrm{y} + \mathrm{b})\Big)\Big(1 - \hat{\mathrm{F}}(\mathrm{b})\Big) \qquad (4.3)$$

where $\hat{\mathrm{F}}(\mathrm{y}) = \mathrm{P}(\mathrm{Y} \leq \mathrm{y})$ is the specified cumulative distribution function which will be fitted to the data above the threshold b. We want to incorporate the tail distribution into the spline model for values above the last knot $\mathrm{xc_N}$.
So by letting $\hat{\mathrm{F}}^{-1}(\mathrm{xc_N}) = \mathrm{b}$ we can write:

$$\mathrm{P}(\mathrm{Y} > y) = \Big(1 - \hat{\mathrm{F}}(\mathrm{y} + \mathrm{b})\Big)\Big(1 - \hat{\mathrm{F}}\Big(\hat{\mathrm{F}}^{-1}(\mathrm{xc_N})\Big)\Big) = \Big(1 - \hat{\mathrm{F}}(\mathrm{y} + \mathrm{b})\Big)(1 - \mathrm{xc_N}) \quad (4.4)$$

Exponential distribution:
One of the distributions that can be used is the exponential one, its cumulative distribution function is:

$$\mathrm{F}(\mathrm{y};\beta) = 1 - \mathrm{e}^{-\frac{\mathrm{y}}{\beta}} \qquad (4.5)$$

By setting (4.5) into (4.4) and renaming $\beta = \mathrm{a_N}$ we get:

$$\mathrm{P}(\mathrm{Y} > y) = \Big(1 - \hat{\mathrm{F}}(\mathrm{y} + \mathrm{b})\Big)(1 - \mathrm{xc_N}) = \left(e^{\frac{-y+b}{\mathrm{a_N}}}\right)(1 - \mathrm{xc_N}) \qquad (4.6)$$

In order to find the percentile function above the threshold we need to solve:

$$P(Y > y) = 1 - u \Leftrightarrow y = b - a_N \log\left(\frac{1-u}{1-xc_N}\right), \text{which means that;}$$

$$\hat{F}^{-1}(u) = b - a_N \log\left(\frac{1-u}{1-xc_N}\right), \text{ if } u > xc_N \quad (4.7)$$

Given (4.7) we can conclude that a spline model with an exponentially distributed tail incorporated will have the following representation:

$$\hat{F}^{-1}(u) = \sum_{j=1}^{N-1} a_j (u - xc_j)_+^3 + u * d_1 \quad \text{if } u \leq xc_N \quad (4.8)$$

$$\hat{F}^{-1}(u) = \sum_{j=1}^{N-1} a_j (xc_N - xc_j)_+^3 + xc_N * d_1 - a_N \log\left(\frac{1-u}{1-xc_N}\right) \quad \text{if } u > xc_N \quad (4.8)$$

The spline model is easy to implement into an optimization procedure and results from a minimization of (4.2) with a spline model with exponential tail (4.8) incorporated are illustrated below.

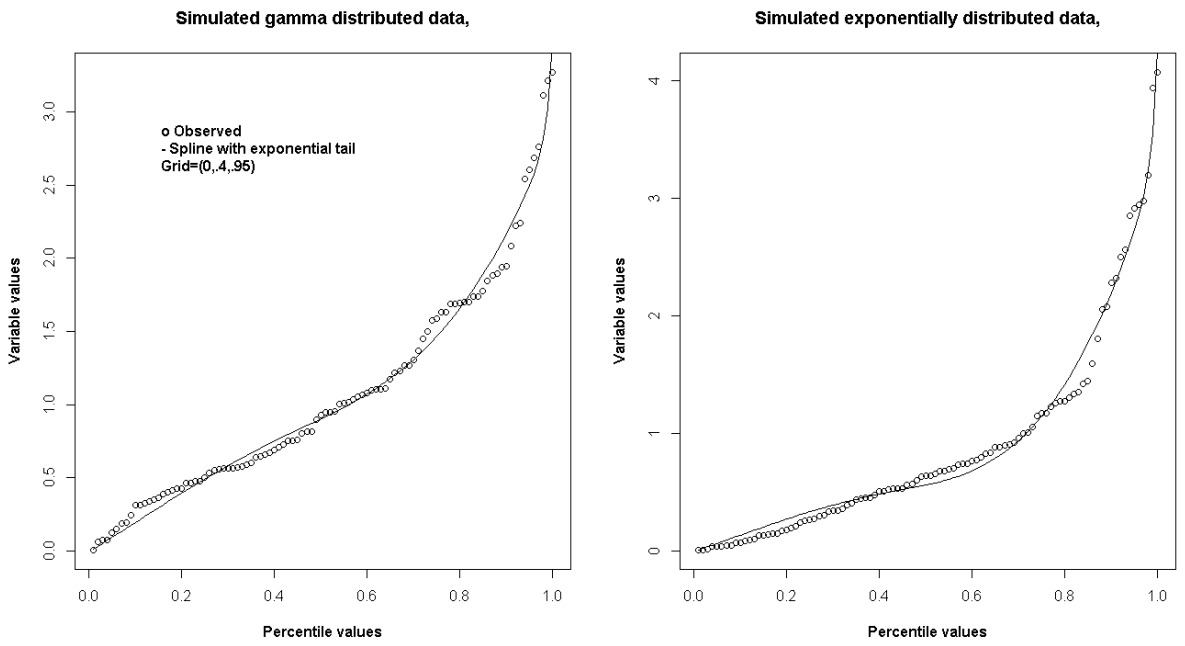

Figure 4.2: Spline model with exponential tail fitted to gamma-distributed data(**left**) and exponentially distributed data(**right**).

The grid was chosen so that only the 5% biggest data points are fitted to the tail distribution. By looking at figure 4.2 we can conclude that incorporating an exponential tail distribution into the spline model (4.8) gives a better fit to the tails of the simulated data then the spline model without a tail distribution(4.1).

Pareto distribution:

Another distribution that has interesting tail properties and which can be used is the Pareto one, its cumulative distribution function is:

$$F(y;\alpha,\beta) = 1 - \left(1 + \frac{y}{\beta}\right)^{-\alpha} \quad (4.9)$$

By setting (4.9) into (4.4) and renaming $\beta = a_N$ we get:

$$P(Y > y) = \left(\left(1 + \frac{y-b}{a_N}\right)^{-\alpha}\right)(1 - xc_N) \quad (4.10)$$

Solving $P(Y > y) = 1 - u$ gives:

$$y = a_N\left(\left(\frac{1-u}{1-xc_N}\right)^{-\frac{1}{\alpha}} - 1\right) + b, \text{ which means that;}$$

$$\hat{F}^{-1}(u) = a_N\left(\left(\frac{1-u}{1-xc_N}\right)^{-\frac{1}{\alpha}} - 1\right) + b, \text{ if } u > xc_N \qquad (4.11)$$

Given (4.11) we can conclude that a spline model with a Pareto distributed tail incorporated will have the following representation:

$$\hat{F}^{-1}(u) = \sum_{j=1}^{N-1} a_j(u - xc_j)_+^3 + u * d_1 \quad \text{if } u \le xc_N \quad (4.12)$$

$$\hat{F}^{-1}(u) = \sum_{j=1}^{N-1} a_j(xc_N - xc_j)_+^3 + xc_N * d_1 + a_N\left(\left(\frac{1-u}{1-xc_N}\right)^{-\frac{1}{\alpha}} - 1\right) \text{ if } u > xc_N \quad (4.12)$$

This spline model is also easily incorporated into an optimization scheme and results from a minimization of (4.2) with a spline model with pareto tail (4.12) are illustrated on the next page.

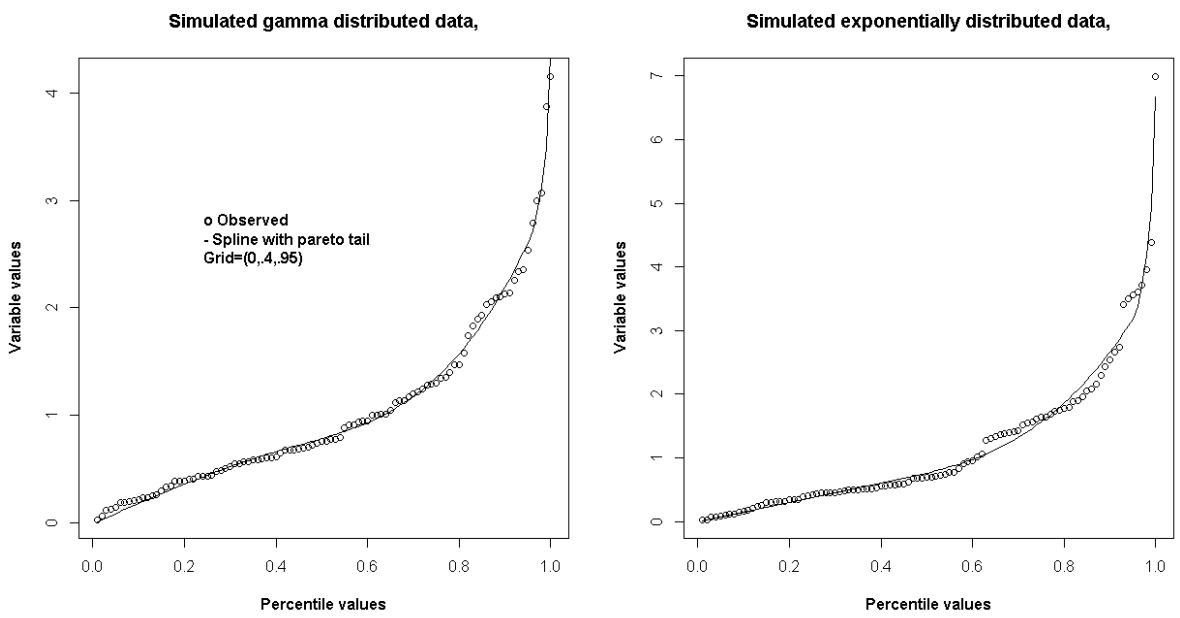

Figure 4.3: Spline model with pareto tail fitted to gamma-distributed data(**left**) and exponentially distributed data(**right**).

By looking at figure 4.3 it can be concluded that also the spline model with a Pareto distributed tail (4.12) has a better fit then the spline model without any tail distribution(4.1). When fitting the spline model with Pareto tail(4.12) to the simulated data α was chosen to be 10. The optimal value for α will be investigated in chapter 4.5.

4.4 Skewness.

Skewness is a measure of symmetry in a distribution; perfectly symmetric distributions like the normal and uniform distribution have a skewness value of 0. One can also say that skewness is a measure of how much a distribution leans to one side of the mean, which is nicely illustrated in the figure below.

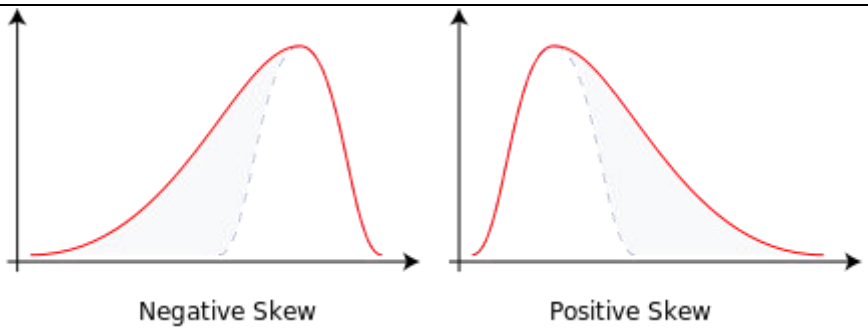

Figure 4.4: Illustration of skewness. If a distribution leans to the right it has a negative skewness(**left**) and the other way around for positive skewness(**right**).

The skewness in a distribution can be calculated through the moments of the distribution:

$$\text{Skewness} = \frac{V_3}{\sigma^3}$$

Where V_3 is the third order moment and σ is the standard deviation. The moments in continuous distributions like the gamma and exponential are calculated by integration:

$$E\{H(y)\} = \int_{-\infty}^{\infty} H(y)\, f(y)\, dy \quad (4.13)$$

Where f(y) is the density distribution function and H(y) a function dependent on the order of the moment which is to be calculated.

$H(y) = y \quad \rightarrow \quad \xi = E(y)$	First order moment
$H(y) = (y-\xi)^2 \quad \rightarrow \quad \sigma^2 = E(y-\xi)^2$	Second order moment
$H(y) = (y-\xi)^3 \quad \rightarrow \quad V_3 = E(y-\xi)^3$	Third order moment

Table 4.1: Illustration of H(y) functions needed to calculate the skewness.

Skewness in gamma distribution

The density function of the gamma distribution which were used to simulate datasets in chapter 4.2 and 4.3 and its corresponding skewness:

$$f(y,\beta) = \frac{y^{\beta-1}e^{-y}}{\Gamma(\beta)} \text{ for } y>0$$

$$\text{skewness}(y) = \frac{2}{\sqrt{\beta}}$$

Skewness in exponential distribution

$$f(y,\beta) = \frac{1}{\beta}e^{-\frac{y}{\beta}} \text{ for } y \geq 0$$

$$\text{skewness}(y) = 2$$

Skewness is an important concept in property insurance because it has to do with the right tail of the distribution used to model claim severity. If a distribution is used which has a lower skewness than the underlying empirical distribution, then the right tail of the distribution is underestimated and effectively the size of the largest claims is underestimated. In other words, choosing the wrong distribution can have a huge financial impact. In the next section I will estimate the skewness in spline models fitted to simulated datasets from distributions with a known skewness value. This is done to determine if spline models are suitable for modeling heavy tailed phenomena such as property insurance claims severity.

4.5 Estimating skewness in spline models

Exactly how well can a spline model estimate the skewness in the distribution of the underlying data? That can be answered by performing experiments with simulated data from a distribution with a known skewness value and see if the estimated skewness value of the fitted spline model is the same as for the underlying distribution in the simulated dataset. In order to estimate the moments for the spline model which again is used to calculate an estimate of the skewness one can use Gauss-Legendre quadrature integration. By using substitution (4.13) can be written as:

$$E\{H(y)\} = \int_0^1 H(F^{-1}(u))du \quad \text{because} \begin{cases} u = F(y) \Leftrightarrow F^{-1}(u) = y \\ du = f(y)dy \end{cases} \quad (4.14)$$

If the percentile function $F^{-1}(u)$ is a spline model with knots $xc_1, \dots, xc_N$ we can write:

$$E\{H(y)\} = \sum_{j=1}^{N} \int_{xc_j}^{xc_{j+1}} H(F^{-1}(u))du = \sum_{j=1}^{N} I_j \quad (4.15)$$

$$\text{where } I_j = \left[\sum_{s=1}^{m} w_s \, H\left(F^{-1}(v_s)\right)\right]\left(xc_{j+1} - xc_j\right)$$

$$\text{with } v_s = xc_j + x_s\left(xc_{j+1} - xc_j\right) \text{ and } xc_{N+1} = 1.$$

The abscissas x_s and weights w_s have been taken from appendix C.3 in Bølviken(2014). These m=10 non-equidistant abscissas and weights have been carefully constructed for integration in the area [0,1]. By using (4.15) and the functions in table 4.1 skewness estimates for the spline models fitted simulated data can be found. R was used to simulate 200 datasets and fit spline models to each of them. The skewness estimates for each spline was calculated and results for the mean and standard deviation of the estimates are illustrated in the tables below for three different sizes on the datasets. When gamma distributed data was simulated β was chosen to be 2 which gives a skewness in the underlying data of 1.414.

Basic spline model:

Table 4.2 shows skewness estimates for a basic spline model (4.1) fitted to simulated gamma distributed data of varying dataset size. The spline model is massively underestimating the skewness in the underlying data which is 1.414. The basic spline model(4.1) was also fitted to simulated exponentially distributed data. Table 4.3 shows that the basic spline model is underestimating the skewness in the underlying data in this case as well. But the estimates are closer to the real values for exponential data. This probably has to do with the tail properties of exponentially and gamma distributed data. The basic spline model does not fit the tails of the data good, and since exponentially distributed data has a smaller tail than gamma distributed data with $\beta = 2$, the spline model is closer to estimating the real skewness value for exponentially distributed data.

Skewness Estimates:	**Mean:**	**Sd:**
n=100	1.063	0.344
n=1000	1.105	0.111
n=10000	1.119	0.035

Table 4.2: Skewness estimates for a spline model(4.1) fitted to gamma distributed data with skewness value 1.414.

Skewness Estimates:	**Mean:**	**Sd:**
n=100	1.673	0.386
n=1000	1.729	0.122
n=10000	1.729	0.041

Table 4.3: Skewness estimates for a spline model(4.1) fitted to exponentially distributed data with skewness value 2.

Spline model with exponential tail:

Table 4.4 and 4.5 shows skewness estimates for a spline model with exponential tail fitted to the same gamma and exponentially distributed data that the basic spline model was fitted to. The two tables show that skewness values are much closer to the skewness value in the underlying data than the skewness estimates for the basic spline model were. The tables shows the same thing as the graphs did earlier in the chapter, incorporating tail distributions in the spline model will make the fit to long tailed data better.

Skewness Estimates:	**Mean:**	**Sd:**
n=100	1.339	0.559
n=1000	1.303	0.173
n=10000	1.305	0.052

Table 4.4: Skewness estimates for a spline model with exponential tail(4.8) fitted to gamma distributed data with skewness value 1.414.

Skewness Estimates:	**Mean:**	**Sd:**
n=100	1.945	0.648
n=1000	1.987	0.224
n=10000	1.982	0.073

Table 4.5: Skewness estimates for a spline model with exponential tail(4.8) fitted to exponentially distributed data with skewness value 2.

Spline model with Pareto tail:

A spline with pareto tail with 10 different values of α was like the two other spline models fitted to 200 simulated gamma distributed and 200 exponentially distributed datasets. Since the results in table 4.2, 4.3, 4.4 and 4.5 showed that there was little uncertainty in the estimates when each dataset had 1000 observations, only datasets of that size was simulated. Skewness estimates for a spline model with Pareto tail are shown in the two tables below.

α	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Mean	1.304	1.324	1.308	1.327	1.312	1.321	1.317	1.318	1.329	1.316
SD	0.169	0.151	0.149	0.165	0.163	0.162	0.162	0.147	0.159	0.173

Table 4.6: Skewness estimates for a spline model with Pareto tail(4.12) fitted to gamma distributed data with skewness value 1.414.

α	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Mean	1.970	2.006	2.043	1.984	1.984	2.019	2.025	1.996	1.979	1.969
SD	0.245	0.252	0.226	0.208	0.230	0.259	0.253	0.236	0.234	0.234

Table 4.7: Skewness estimates for a spline model with Pareto tail(4.12) fitted to exponentially distributed data with skewness value 2.

By looking at table 4.6 and 4.7 we see that the skewness estimates are pretty much the same regardless of the α value in the spline model. We also see that the skewness estimates are pretty much the same as for the spline model with an exponential tail distribution. Although we know from theory that the Pareto distribution has more interesting tail properties than the exponential one, it can be argued that the first tail distribution used to fit a spline model to a dataset should be exponential because it estimates skewness just as well. After all the Pareto distribution converges to an exponential one when $\alpha \to \infty$.

4.6 Monotonicity in spline models

There's no mathematical justification for why the spline models in chapter 3 and 4 are monotone. In order to obtain a definitive monotone model one can place the following conditions on $c_1, \dots, c_N$; $c_1 > 0$ and $c_s > \frac{-c_1(xc_{s+1}-xc_1) - \dots - c_{s-1}(xc_{s+1}-xc_{s-1})}{xc_{s+1}-xc_s}$ for $s = 2, \dots, N$. where $x_{N+1} = 1$. Optimization is done best when $c_1, \dots, c_N$ are transformed to $v_1, \dots, v_N$ through $v_1 = \log(c_1)$ and $v_s = \log\left(c_s + \frac{c_1(xc_{s+1}-xc_1) + \dots + c_{s-1}(xc_{s+1}-xc_{s-1})}{xc_{s+1}-xc_s}\right)$ for $s = 2, \dots, N$. The monotonicity constraints were implemented while working on the thesis, but not in the procedures which made the results shown in this thesis. The unconstrained procedures always proved to be monotone, and implementing the constraints only lead a longer R-code and more or less the same results. However, if an unconstrained procedure leads to a non-monotone result when trying to model phenomena, the monotonicity constraints are a good tool to have.

Chapter 5

Claims severity modelling II

5.1 Introduction

The results in chapter 4 indicates that spline models with exponentially or pareto distributed tails estimate the skewness in the underlying simulated data quite well. But do they do the same for real datasets of insurance claims? In this chapter results will be shown from fitting a spline model with an exponentially distributed tail to three datasets of real insurance claims of varying size and skewness value to check how well the model tackles different scenarios. I will also estimate insurance reserves based on the datasets using a spline model and the empirical data. The hypothesis is that the reserves will be higher when using a spline model since it can estimate claims larger than the ones already observed, unlike sampling from the empirical distribution.

5.2 Spline model fitted to empirical data

This chapter focuses on results from fitting a spline model with exponential tail to three datasets of insurance claims.

Danish fire claims:

This a dataset of Danish fire insurance claims over 1 million Danish kroner. The claims occurred in the period 1980-1990 and would have been much larger if they happened today. There are 2167 claims in the dataset, the mean of the claims is 3.39 million and the standard deviation is 8.51 million. The skewness in the empirical distribution is huge, 18.7. In Figure 5.1 (figure below) results from fitting a spline model with an exponential tail to the Danish fire claims is illustrated. Looking at the graph to the left in figure 5.1 one could easily believe that the model fits the data well since it's hard to see the fit to the lower values, but by plotting the values on a log-scale one can see that the model does not fit the data well, especially for the lower claims. The fit did not improve when the author tried different grid-structures.

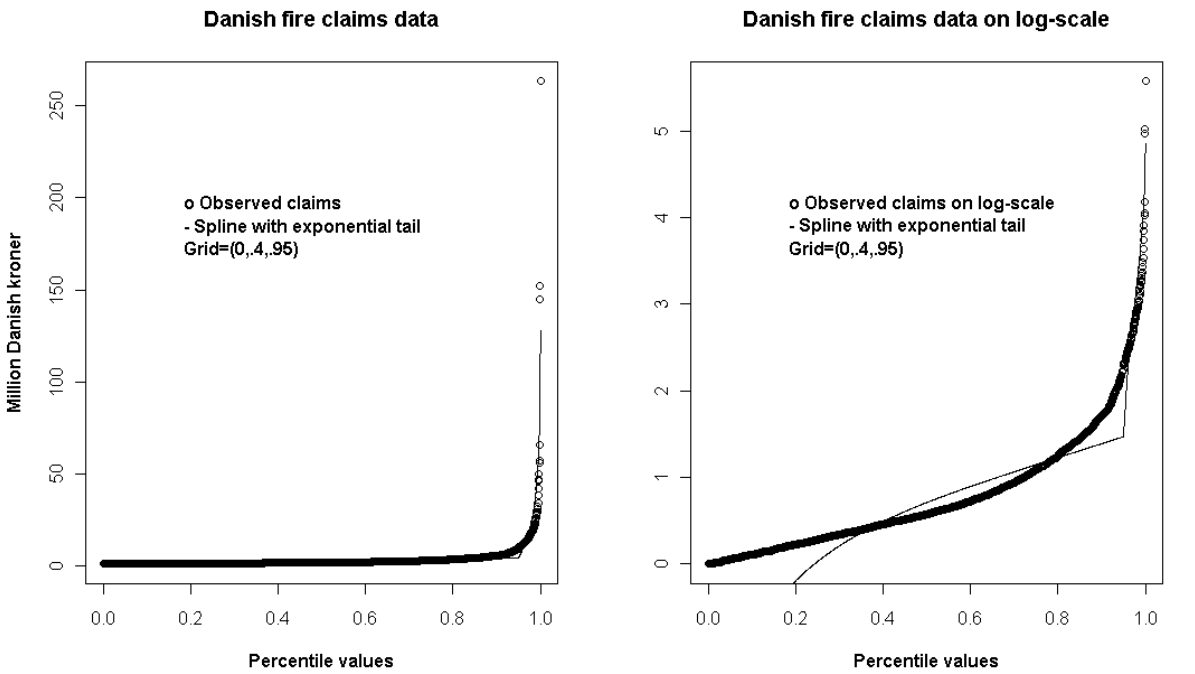

Figure 5.1: Spline model with exponential tail fitted to Danish fire claims data(**Left**) and Danish fire claims data on log-scale.

Belgian fire claims:

This dataset contains 60 observations of fire claims in Belgium. The mean is 20.89 million euro's and the standard deviation is 21.45 million euro's, skewness is 1.49. The spline model fits the Belgian fire claims almost perfectly.

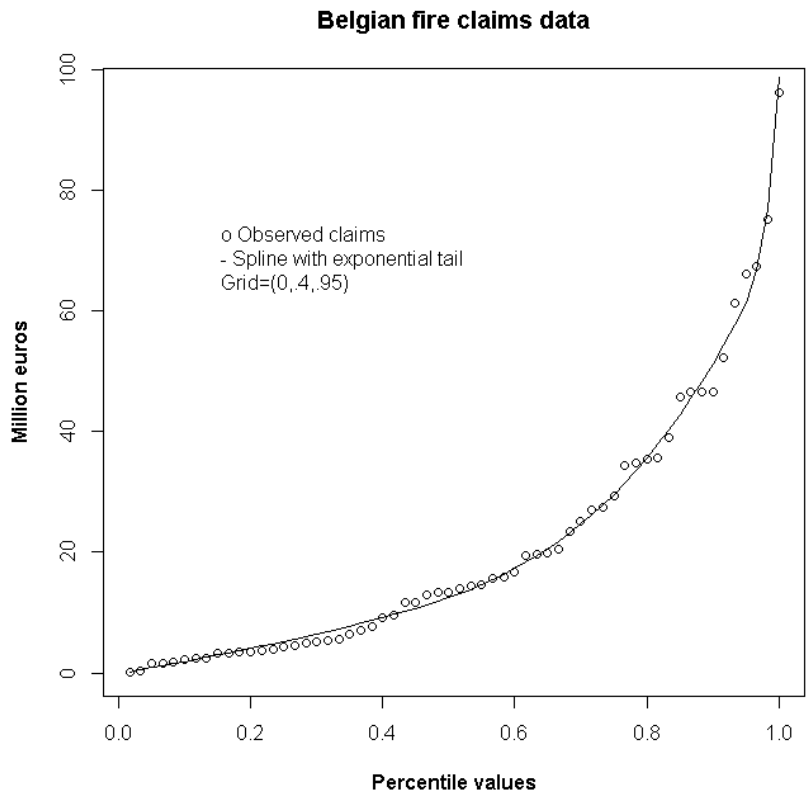

Figure 5.2: Spline model with exponential tail fitted to Belgian fire claims data.

US hurricane claims:

The dataset contains 35 observations of total claims due to hurricanes in the US for the period 1949-80. The claims are in million US dollars and the mean is 204 million while the standard deviation is 330 million. The skewness of the claims is 2.91. Figure 5.2 illustrates the fit of the spline model to the hurricane data. The model fits the tail of the data well, but underestimates the smaller claims and overestimates the midsized claims.

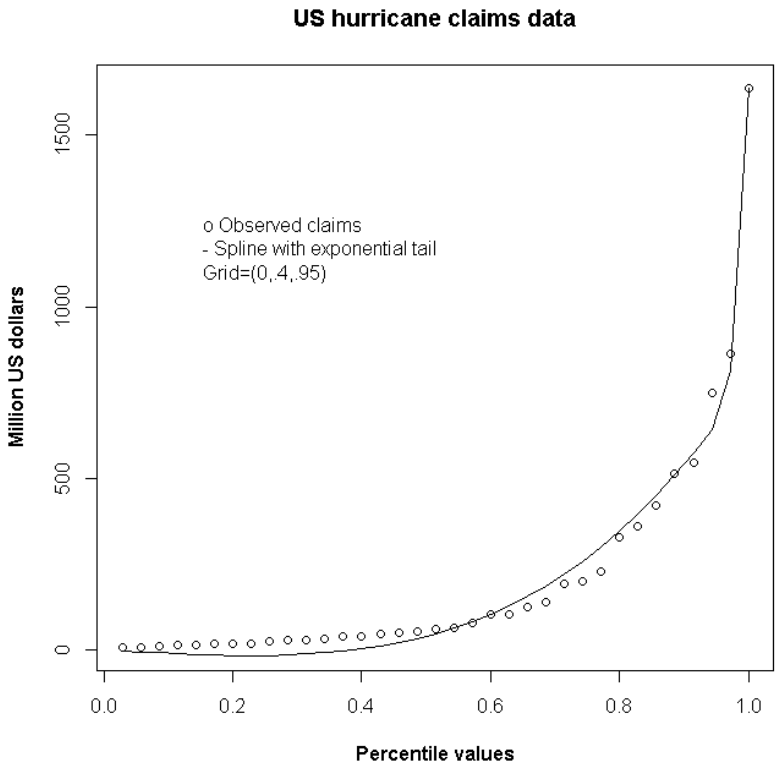

Figure 5.3: Spline model with exponential tail fitted to US hurricane claims data.

5.3 Analysis of insurance reserves, empirical datasets.

Insurance companies are by law required to set aside a reserve in order to meet future obligations for a given period. A requirement could be that this reserve should be so high that it will cover the future obligations in 99 out of a 100 times in that given period. A way to calculate a reserve like that is to decide a claims frequency for that period and sample claims from either an empirical or parametric distribution. The sum of the claims equals the reserve. In order to find an estimate for the reserve that will cover the sum of the claims in 99 out of a 100 times i.e the 99th percentile, a reserve X needs to simulated say m times: $X_i = \sum_{j=1}^{n} y_j$ for $i = 1, \dots, m$, where y is a sampled claim. By ordering the simulated reserves by size $X_{(1)} \geq X_{(2)} \geq .. \geq X_{(m)}$ we can easily estimate the 99th percentile by finding the reserve that is bigger than 99% of the other reserves. If 1000 simulations of a reserve is done, the 99th percentile will be $X_{(990)}$. Say we want to calculate reserves for the three datasets described earlier in the chapter for the same number of years into the future as the number of years the data was collected from. Given that there is no inflation, no changes in exposure and the claims frequency stays the same, we can calculate reserves $X_i = \sum_{j=1}^{n} y_j$ for $i = 1, \dots, m$ where n is equal to the size of the empirical datasets. Mean and percentiles of reserves when m=1000 have been calculated for all three empirical datasets using empirical sampling and sampling from a spline model with exponential tail fitted to the empirical data, the results are shown below.

		Percentiles	
	Mean	95 %	99 %
Empirical	7 330	8 023	8 358
Spline	7 078	7 698	7 977

Table 5.1: Reserve estimates computed from empirical sampling and sampling from spline model fitted to Danish fire claims data.

Table 5.1 shows that reserves estimated based on sampling from a spline model has lower values for all three measures than reserves estimated based on empirical sampling. Especially the differences in the 99th percentile estimates are huge. Estimating reserves from spline model sampling for Danish fire claims can potentially lead to underestimation of future liabilities.

		Percentiles	
	Mean	95 %	99 %
Empirical	1 250	1 524	1 622
Spline	1 302	1 610	1 721

Table 5.2: Reserve estimates computed from empirical sampling and sampling from spline model fitted to Belgian fire claims data.

Table 5.2 tells a different story than table 5.1. Here it's the reserve measures based on sampling from a spline model that is bigger than the ones based on empirical sampling. This probably has to do with the fit the spline model had to the two datasets. The spline model fit the Belgian fire claims data almost perfectly, and it can predict claims bigger than the ones in the dataset it has been fitted to, which might be beneficial when predicting future reserves. The spline model fit to the Danish fire claims data was not good, both the smallest and largest claims were underestimated. It might be plausible to say that spline model is more effectively used on smaller datasets.

		Percentiles	
	Mean	95 %	99 %
Empirical	7 206	10 769	12 614
Spline	8 335	14 002	18 426

Table 5.3: Reserve estimates computed from empirical sampling and sampling from spline model fitted to US hurricane claims data.

Looking at table 5.3 one can see that there is a huge difference between the 99th percentile reserves estimates based on the US hurricane data. This can be explained by the extreme tail properties of the spline model. The smaller the dataset, the bigger the possibility that a sample from a spline model will be bigger than the biggest observation in the underlying dataset. This property might make the spline model especially useful when modelling natural catastrophe claims.

5.4 Analysis of insurance reserves, simulated datasets.

Results in chapter 5.3 showed that sampling from spline models resulted in higher reserves than empirical sampling when dealing with relatively small datasets. For the Danish fire claims data it is safe to say that empirical sampling would be preferred when estimating reserves. This chapter will also focus on the estimation of reserves and the 99th percentiles of these. But the twist from the preceding chapter is that the reserves will be based on simulated pareto and gamma distributed data. Parameters for the two parametric models are chosen so that the simulated pareto distributed data is long tailed while the gamma distributed data is relatively small tailed. By estimating reserves based on datasets of varying size the hypothesis is that the estimates based on sampling from a spline model and empirical sampling will be closer to each other when the underlying data is short tailed and contains more observations.

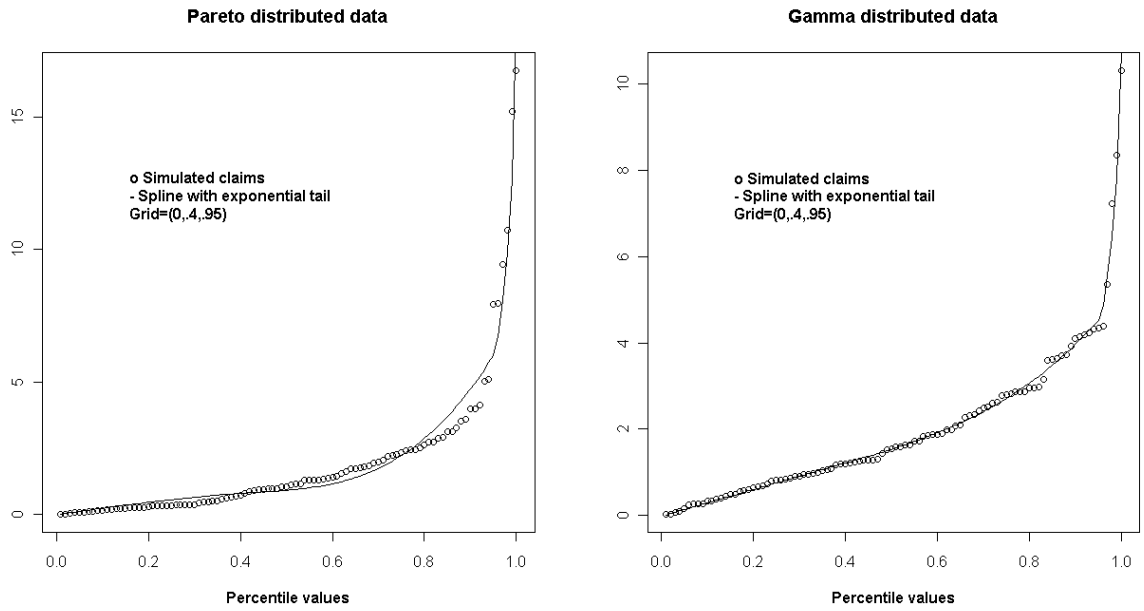

Figure 5.4: Spline model with exponential tail fitted to simulated pareto distributed data(**Left**) and simulated gamma distributed data(**Right**).

Figure 5.4 illustrates that the spline model with exponentially distributed tail fits pareto and gamma distributed simulated data well. Each dataset contains 100 observations and the pareto distributed data is a lot more long tailed then the gamma distributed data. Below are results from estimating reserves based on simulated data from the two the two distributions.

		Mean	95 %	99 %
100 claims in dataset	Empirical	238	266	276
	Spline	239	269	284
500 claims in dataset	Empirical	1020	1073	1095
	Spline	1012	1066	1087
1000 claims in dataset	Empirical	1949	2019	2041
	Spline	1939	2012	2046

Table 5.4: Reserve estimates computed from empirical sampling and sampling from spline model fitted to simulated gamma distributed claims data with varying sample size.

The results in table 5.4 are quite similar for the two sampling methods, the little difference there is for the smallest dataset is due to the tail properties of the spline model.

		Mean	95 %	99 %
100 claims in dataset	Empirical	209	266	292
	Spline	224	297	335
500 claims in dataset	Empirical	989	1147	1230
	Spline	1046	1191	1272
1000 claims in dataset	Empirical	1659	1804	1856
	Spline	1680	1820	1866

Table 5.5: Reserve estimates computed from empirical sampling and sampling from spline model fitted to simulated pareto distributed claims data with varying sample size.

There is a bigger difference in estimates in table 5.5 than in table 5.4. The difference in the 99th percentile when the dataset contains 100 observations is as big as 15%. The difference almost vanishes when the datasets are larger, but not to the same degree as it does for the gamma distributed data. Based on these results one could argue that a spline model would be preferred when estimating reserves based on long tailed data with around 500 observations or less. Empirical sampling won't be able to catch the tails for such datasets, but as the sample size increases it might be sufficient to use empirical sampling, perhaps with an incorporated tail distribution.

6 Concluding remarks:

Once the algorithms for the spline models are set up, it's easy and fast to estimate parameters for different datasets. Not having to find a suitable distribution to the dataset saves a lot of time and potentially a lot of money in software license fees for insurance companies. It's also easy to incorporate link-functions, constant and linear terms. The cubic and quadratic spline models introduced in chapter 3 didn't prove to be better than the well established Gompertz-Makeham model. But they weren't a lot worse either, and while it can be claimed that the Gompertz-Makeham model has few areas of usage besides mortality modelling, the spline models are based on a general technique that can be used in a lot of other areas. And one of those areas is the modelling of claims severity which was covered in chapter 4 and 5. It was shown in chapter 4 that spline models with incorporated tail distributions estimated the skewness in underlying distributions well. That is a highly valuable property for a model to have when dealing with long tailed datasets, which is common in the insurance industry. In chapter 5 a spline model with exponential tail was fitted to three empirical datasets of different sizes and insurance reserves was estimated through Monte Carlo-simulation for each dataset. These reserves were compared to reserves made by empirical sampling. The comparison showed that the spline model is preferred over an empirical model when computing reserves based on smaller datasets. This has to do with the fact that no claim can be bigger than the ones already seen when using empirical sampling, which might be unfortunate when dealing with smaller datasets. When working with large datasets like the Danish fire claims it´s not that likely that a larger claim than already observed will occur. It might therefore be sufficient to adjust for inflation and use empirical sampling when estimating insurance reserves for larger datasets. The results in chapter 5.4 told the same story, and they even showed that empirical sampling might be an option even for smaller datasets, as long as the data is short tailed. For long tailed datasets the spline models were superior and would be a preferred default model for smaller datasets with all sorts empirical distributions. A theme that hasn't been highlighted in this thesis is the codes used to produce the results shown in the previous chapters. All codes were written in R and the aim was to write them as compact and general as possible so that they could be implemented into automated processes and standard software. The appendix contains all empirical datasets and codes used to create the results in this

thesis. I believe that this code can easily be implemented into an automatic process. The reader is encouraged to replicate the results in this thesis by using the codes and datasets in the appendix.

7 References

M. R. Steenbergen. (2006). A primer of maximum likelihood programming in R. Lecture note for the course poli 270 taught at Department of political science, UCSA. San Diego

Human mortality database (HMD). http://www.mortality.org/.

E. Bølviken. (2014). Computation and Modelling in Insurance and Finance. Cambridge university press. Cambridge

E. Pitacco. (2004). From Halley to Frailty: A review of survival models for actuarial calculations. Giornale dell'Istituto Italiano degli Attuari, LXVII (1-2): 17-47.

J. A. Nelder and P. McCullagh. (1989). Generalized linear Models, Second edition. Chapman and Hall. London.

A Appendix

A.1 R-codes used for results in chapter 3

A.1.1 Figure 3.1

```
x= datamalenorway2011[,1]
Y=ceiling(datamalenorway2011[,2])
N= datamalenorway2011[,3]
par(mfrow=c(1,2),font=2)
plot(x,N,ylab="",xlab="Age",main="Age distribution in real dataset",font=2, font.lab=2)

J=100000
n0=exp(-0.05*abs(x-50))
c=J/sum(n0)
n=ceiling(c*n0)
plot(x,n,ylab="",xlab="Age",main="Age distribution in simulated dataset",font=2, font.lab=2)
legend(60,2500,"J=100000",bty="n")
```

A.1.2 Figure 3.2

```
y=1:length(x)*0
for (k in 1:length(x))
{y[k]=rbinom(1,n[k],Y[k]/N[k])}

minusll=function(s,y,n,M,A,xc,nc)
{a_0=s[1]
for (k in 2:(nc+1))
{c[k-1]=atan(s[k])*(2*A/pi)}
lq=a_0+c%*%M
-sum(y*lq-n*log((1+exp(lq))))}

derivatives=function(s,y,n,M,A,xc,nc)
{a_0=s[1]
for (k in 2:(nc+1))
{c[k-1]=atan(s[k])*(2*A/pi)}
lq=a_0+c%*%M
c(-sum(y-n*exp(lq)/(1+exp(lq))),
-sum(y*(1/(1+s[2]**2))*(2*A/pi)*(pmax(xc[1]-x,0)**2)-((n*(1/(1+s[2]**2))*(2*A/pi)*(pmax(xc[1]-
x,0)**2))/(1+exp(-lq)))),
-sum(y*(1/(1+s[3]**2))*(2*A/pi)*(pmax(xc[2]-x,0)**2)-((n*(1/(1+s[3]**2))*(2*A/pi)*(pmax(xc[2]-
x,0)**2))/(1+exp(-lq)))),
-sum(y*(1/(1+s[4]**2))*(2*A/pi)*(pmax(xc[3]-x,0)**2)-((n*(1/(1+s[4]**2))*(2*A/pi)*(pmax(xc[3]-
x,0)**2))/(1+exp(-lq)))))
}

splinefit=function(x,y,n,xc,A)
{nc=length(xc)
nx=length(x)
M=pmax(xc-t(matrix(x,nx,nc)),0)**2
s=rep(-.1,nc+1)
o=optim(s,minusll,derivatives,method="BFGS",y,n,M,A,xc,nc,control=list(maxit=8000))
a_0=o$par[1]
for (k in 2:(nc+1))
```

```
{c[k-1]=atan(o$par[k])*(2*A/pi)}
lq=a_0+c%*%M
q=1/(1+exp(-lq))
list(q=q,AIC=(-2)*(-
optim(s,minusll,derivatives,method="BFGS",y,n,M,A,xc,nc,control=list(maxit=8000))$value)+2*(nc+1))}
q_quad=splinefit(x,y,n,xc=c(45,90,105),A=0.1)$q
AIC_quad=splinefit(x,y,n,xc=c(45,90,105),A=0.1)$AIC

par(mfrow=c(1,2),font=2)
plot(x,y/n,ylab="",xlab="Age",main="Annual mortality probabilities",font=2, font.lab=2)
legend(x[3],0.6, c("o Observed","- Quadratic spline","J=100000","Grid=(45,90,105)"),bty="n")
lines(x,q_quad)

plot(x,log(y/n),ylab="",xlab="Age",main="Annual mortality probabilities on log-scale",font=2, font.lab=2)
lines(x,log(q_quad))
```

A.1.3 Figure 3.3

```
minusll=function(s,y,n,M,A,xc,nc)
{a_0=s[1]
for (k in 2:(nc+1))
{c[k-1]=atan(s[k])*(2*A/pi)}
lq=a_0+c%*%M
-sum(y*lq-n*log((1+exp(lq))))}

derivatives=function(s,y,n,M,A,xc,nc)
{a_0=s[1]
for (k in 2:(nc+1))
{c[k-1]=atan(s[k])*(2*A/pi)}
lq=a_0+c%*%M
c(-sum(y-n/(1+exp(-lq))),
-sum(y*(1/(1+s[2]**2))*(2*A/pi)*(pmax(xc[1]-x,0)**3)-((n*(1/(1+s[2]**2))*(2*A/pi)*(pmax(xc[1]-
x,0)**3))/(1+exp(-lq)))),
-sum(y*(1/(1+s[3]**2))*(2*A/pi)*(pmax(xc[2]-x,0)**3)-((n*(1/(1+s[3]**2))*(2*A/pi)*(pmax(xc[2]-
x,0)**3))/(1+exp(-lq)))),
-sum(y*(1/(1+s[4]**2))*(2*A/pi)*(pmax(xc[3]-x,0)**3)-((n*(1/(1+s[4]**2))*(2*A/pi)*(pmax(xc[3]-
x,0)**3))/(1+exp(-lq)))))
}

splinefit=function(x,y,n,xc,A)
{nc=length(xc)
nx=length(x)
M=pmax(xc-t(matrix(x,nx,nc)),0)**3
s=rep(-.1,nc+1)
o=optim(s,minusll,derivatives,method="BFGS",y,n,M,A,xc,nc,control=list(maxit=8000))
a_0=o$par[1]
for (k in 2:(nc+1))
{c[k-1]=atan(o$par[k])*(2*A/pi)}
lq=a_0+c%*%M
q=1/(1+exp(-lq))
list(q=q,AIC=(-2)*(-
optim(s,minusll,derivatives,method="BFGS",y,n,M,A,xc,nc,control=list(maxit=8000))$value)+2*(nc+1))}
q_cubic=splinefit(x,y,n,xc=c(95,100,105),A=0.1)$q
AIC_cubic=splinefit(x,y,n,xc=c(95,100,105),A=0.1)$AIC

par(mfrow=c(1,2),font=2)
plot(x,y/n,ylab="",xlab="Age",main="Annual mortality probabilities",font=2, font.lab=2)
```

```
legend(x[3],0.6, c("o Observed","- Cubic spline","J=100000","Grid=(95,100,105)"),bty="n")
lines(x,q_cubic)

plot(x,log(y/n),ylab="",xlab="Age",main="Annual mortality probabilities on log-scale")
lines(x,log(q_cubic))
```

A.1.4 Figure 3.4

```
minusgompmakll=function(s,x,n,y)
{t=exp(s)
q=1-exp(-t[1]-t[2]*(exp(t[3])-1)*exp(t[3]*x)/t[3])
-sum(y*log(q)+(n-y)*log(1-q))}

gompmakfit=function(x,n,y)
{s=-c(8,9,2.5)
o=optim(s,minusgompmakll,"BFGS",x,n,y)
t=exp(o$par)
q=1-exp(-t[1]-t[2]*(exp(t[3])-1)*exp(t[3]*x)/t[3])
list(q=q,AIC=(-2)*(-optim(s,minusgompmakll,"BFGS",x,n,y)$value)+2*length(s))}
q_GM=gompmakfit(x,n,y)$q
AIC_GM=gompmakfit(x,n,y)$AIC

par(mfrow=c(1,2),font=2)
plot(x,y/n,ylab="",xlab="Age",main="Annual mortality probabilities",font=2, font.lab=2)
legend(x[1],0.5, c("o Observed","- Gompertz-Makeham","J=100000"),bty="n")
lines(x,q_GM)

plot(x,log(y/n),ylab="",xlab="Age",main="Annual mortality probabilities on log-scale",font=2, font.lab=2)
lines(x,log(q_GM))
```

A.1.4 Table 3.1

```
list("AIC_GM",AIC_GM,"AIC_quad",AIC_quad,"AIC_cubic",AIC_cubic)
```

A.2 R-codes used for results in chapter 4

A.2.1 Figure 4.1

```
spline=function(s,y,M,xc,N,u)
{a_0=s[1]
for (k in 2:(N+1))
{c[k-1]=s[k]}
sum((a_0*u+c%*%M-y)**2)}

splinefit=function(y,xc)
{N=length(xc)
u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(u,length(y),N))-xc,0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)
a_0=o$par[1]
for (k in 2:(N+1))
{c[k-1]=o$par[k]}
list(F=a_0*u+c%*%M)}

par(mfrow=c(1,2),font=2)
y=sort(rgamma(100,2)/2)
x=1:length(y)/length(y)
plot(x,y,main="Simulated gamma distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.8,.95))$F)
legend(.1,3, c("o Observed","- Spline","Grid=(0,.8,.95)"),bty="n")

y=sort(rexp(100))
plot(x,y,main="Simulated exponentially distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.8,.95))$F)
```

A.2.2 Figure 4.2

```
spline=function(s,y,M,xc,N,u)
{sum((s[1]*pmin(u,xc[N])+s[2:N]%*%M-s[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))-y)**2)}

splinefit=function(y,xc)
{N=length(xc)
u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(pmin(u,xc[N]),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)
list(F=o$par[1]*pmin(u,xc[N])+o$par[2:N]%*%M-o$par[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N])))}

par(mfrow=c(1,2),font=2)
y=sort(rgamma(100,2)/2)
x=1:length(y)/length(y)
plot(x,y,main="Simulated gamma distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,3, c("o Observed","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")

y=sort(rexp(100))
plot(x,y,main="Simulated exponentially distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
```

A.2.3 Figure 4.3

```
spline=function(s,y,M,xc,alpha,u,N)
{sum((s[2:N]%*%M+s[1]*pmin(xc[N],u)+s[N+1]*(pmax(((1-u)/(1-xc[N]))**(-1/alpha),1)-1)-y)**2)}

splinefit=function(y,xc)
{alpha=10
N=length(xc)
u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(pmin(xc[N],u),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,alpha,u,N)
list(F=o$par[2:N]%*%M+o$par[1]*pmin(xc[N],u)+o$par[N+1]*(pmax(((1-u)/(1-xc[N]))**(-1/alpha),1)-
1))}

par(mfrow=c(1,2),font=2)
y=sort(rgamma(100,2)/2)
x=1:length(y)/length(y)
plot(x,y,main="Simulated gamma distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,3, c("o Observed","- Spline with pareto tail","Grid=(0,.4,.95)"),bty="n")

y=sort(rexp(100))
plot(x,y,main="Simulated exponentially distributed data,",xlab="Percentile values",ylab="Variable
values",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
```

A.2.4 Table 4.2 and 4.3

```
n=c(100,1000,10000)
mean_skew=sd_skew=1:length(n)*0
for (l in 1:length(n))
{m=200
skew_spline=1:m*0
for (i in 1:m)
{xc=c(0,.8,.95)
N=length(xc)
y=sort(rexp(n[l])) #Change to rgamma(n,alpha)/2 for gamma-distruted simulations

spline=function(s,y,M,N,u)
{sum((s[1]*u+s[2:(N+1)]%*%M-y)**2)}

splinefit=function(y,xc,N)
{u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(u,length(y),N))-xc,0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,N,u)
list(o=o$par)}

o=splinefit(y,xc,N)$o
w=c(.033336,.074726,.109543,.134633,.147762,.147762,.134633,.109543,.074726,.033336)
x=c(.013047,.067468,.160295,.283302,.425563,1-.425563,1-.283302,1-.160295,1-.067468,1-.013047)

moment_one = function(u)
{sum(o[2:(N+1)]*(pmax(u-xc,0)**3))+o[1]*u}

calculation = function(func,x,w,xc,o,N)
{XC=append(xc,1)
I=1:N*0
J=1:length(x)*0
for (s in 1:N)
{for (i in 1:length(x))
{J[i]=w[i]*func(XC[s]+x[i]*(XC[s+1]-XC[s]))*(XC[s+1]-XC[s])}
I[s]=sum(J)}
sum(I)}

mean=calculation(func=moment_one,x,w,xc,o,N)

moment_two = function(u)
{(sum(o[2:(N+1)]*(pmax(u-xc,0)**3))+o[1]*u-mean)**2}

moment_three = function(u)
{(sum(o[2:(N+1)]*(pmax(u-xc,0)**3))+o[1]*u-mean)**3}

skew_spline[i]=calculation(func=moment_three,x,w,xc,o,N)/calculation(func=moment_two,x,w,xc,o,N)**1.
5}
mean_skew[l]=mean(skew_spline)
sd_skew[l]=sd(skew_spline)}
mean_skew
sd_skew
```

A.2.5 Table 4.4 and 4.5

```
n=c(100,1000,10000)
mean_skew=sd_skew=1:length(n)*0
for (l in 1:length(n))
{m=200
skew_spline=1:m*0
for (i in 1:m)
{xc=c(0,.4,.95)
N=length(xc)
y=sort(rgamma(n[l],2)/2)#Change to rexp(n) for exponentially distributed simulations

spline=function(s,y,M,xc,N,u)
{sum((s[1]*pmin(u,xc[N])+s[2:N]%*%M-s[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))-y)**2)}

splinefit=function(y,xc,N)
{u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(pmin(u,xc[N]),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)
list(o=o$par)}

c=splinefit(y,xc,N)$o
w=c(.033336,.074726,.109543,.134633,.147762,.147762,.134633,.109543,.074726,.033336)
x=c(.013047,.067468,.160295,.283302,.425563,1-.425563,1-.283302,1-.160295,1-.067468,1-.013047)

calculation = function(func,x,w,xc,c,N)
{XC=append(xc,1)
I=1:N*0
J=1:length(x)*0
for (s in 1:N)
{for (i in 1:length(x))
{J[i]=w[i]*func(XC[s]+x[i]*(XC[s+1]-XC[s]))*(XC[s+1]-XC[s])}
I[s]=sum(J)}
sum(I)}

moment_one = function(u)
{sum(c[2:N]*(pmax(min(u,xc[N])-xc[1:(N-1)],0)**3))-c[N+1]*log((1-max(u,xc[N]))/(1-
xc[N]))+c[1]*min(u,xc[N])}

mean=calculation(func=moment_one,x,w,xc,c,N)

moment_two = function(u)
{(sum(c[2:N]*(pmax(min(u,xc[N])-xc[1:(N-1)],0)**3))-c[N+1]*log((1-max(u,xc[N]))/(1-
xc[N]))+c[1]*min(u,xc[N])-mean)**2}

moment_three = function(u)
{(sum(c[2:N]*(pmax(min(u,xc[N])-xc[1:(N-1)],0)**3))-c[N+1]*log((1-max(u,xc[N]))/(1-
xc[N]))+c[1]*min(u,xc[N])-mean)**3}

skew_spline[i]=calculation(func=moment_three,x,w,xc,c,N)/calculation(func=moment_two,x,w,xc,c,N)**1.
5}
mean_skew[l]=mean(skew_spline)
sd_skew[l]=sd(skew_spline)}
mean_skew
sd_skew
```

A.2.6 Table 4.6 and 4.7

```
alphav=6:15
skew_mean=skew_sd=1:length(alphav)*0
for (k in 1:length(alphav))
{m=200
skew_spline=1:m*0
for (i in 1:m)
{xc=c(0,.4,.95)
N=length(xc)
y=sort(rgamma(1000,2)/2) #Change to rexp(1000) for exponentially distributed simulations
alpha=alphav[k]

spline=function(s,y,M,xc,alpha,u,N)
{sum((s[2:N]%*%M+s[1]*pmin(xc[N],u)+s[N+1]*(pmax(((1-u)/(1-xc[N]))**(-1/alpha),1)-1)-y)**2)}

splinefit=function(y,xc,alpha,N)
{u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(pmin(xc[N],u),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y=y,M=M,u=u,N=N,alpha=alpha,xc=xc)
list(o=o$par)}

c=splinefit(y,xc,alpha,N)$o
w=c(.033336,.074726,.109543,.134633,.147762,.147762,.134633,.109543,.074726,.033336)
x=c(.013047,.067468,.160295,.283302,.425563,1-.425563,1-.283302,1-.160295,1-.067468,1-.013047)

calculation = function(func,x,w,xc,c,N)
{XC=append(xc,1)
I=1:N*0
J=1:length(x)*0
for (s in 1:N)
{for (i in 1:length(x))
{J[i]=w[i]*func(XC[s]+x[i]*(XC[s+1]-XC[s]))*(XC[s+1]-XC[s])}
I[s]=sum(J)}
sum(I)}

moment_one = function(u)
{sum(c[2:N]*(pmax(min(xc[N],u)-xc[1:(N-1)],0)**3))+c[1]*min(xc[N],u)+c[N+1]*(max(((1-u)/(1-
xc[N]))**(-1/alpha),1)-1)}

mean=calculation(func=moment_one,x,w,xc,c,N)

moment_two = function(u)
{(sum(c[2:N]*(pmax(min(xc[N],u)-xc[1:(N-1)],0)**3))+c[1]*min(xc[N],u)+c[N+1]*(max(((1-u)/(1-
xc[N]))**(-1/alpha),1)-1)-mean)**2}

moment_three = function(u)
{(sum(c[2:N]*(pmax(min(xc[N],u)-xc[1:(N-1)],0)**3))+c[1]*min(xc[N],u)+c[N+1]*(max(((1-u)/(1-
xc[N]))**(-1/alpha),1)-1)-mean)**3}

skew_spline[i]=calculation(func=moment_three,x,w,xc,c,N)/calculation(func=moment_two,x,w,xc,c,N)**1.
5}
skew_mean[k]=mean(skew_spline)
skew_sd[k]=sd(skew_spline)}
skew_mean
skew_sd
```

A.3 R-codes used for results in chapter 5

A.3.1: Figure 5.1, 5.2, 5.3 and 5.4

```
spline=function(s,y,M,xc,N,u)
{sum((s[1]*pmin(u,xc[N])+s[2:N]%*%M-s[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))-y)**2)}

splinefit=function(y,xc)
{N=length(xc)
u=(1:length(y)-0.5)/length(y)
M=pmax(t(matrix(pmin(u,xc[N]),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)
list(F=o$par[1]*pmin(u,xc[N])+o$par[2:N]%*%M-o$par[N+1]*log((1-pmax(u,xc[N]))/(1-
xc[N])),o=o$par)}

par(mfrow=c(1,2),font=2)
y=sort(danishfire)
x=1:length(y)/length(y)
plot(x,y,main="Danish fire claims data",xlab="Percentile values",ylab="Million Danish kroner",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,max(y)*.8,c("o Observed claims","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")
plot(x,log(y),main="Danish fire claims data on log-scale",xlab="Percentile values",ylab="",font.lab=2)
lines(x,log(splinefit(y,xc=c(0,.4,.95))$F))
legend(.1,max(log(y))*.8, c("o Observed","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")
graphics.off()

y=sort(hurricane)
x=1:length(y)/length(y)
plot(x,y,main="US hurricane claims data",xlab="Percentile values",ylab="Million US dollars",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,max(y)*.8,c("o Observed claims","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")

y=sort(belgianfire)
x=1:length(y)/length(y)
plot(x,y,main="Belgian fire claims data",xlab="Percentile values",ylab="Million euros",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,max(y)*.8,c("o Observed claims","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")

par(mfrow=c(1,2),font=2)
y=sort(2*(runif(100)**(-1/2)-1))
x=1:length(y)/length(y)
plot(x,y,main="Pareto distributed data",xlab="Percentile values",ylab="",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,max(y)*.8,c("o Simulated claims","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")
y=sort(rgamma(100,2))
plot(x,y,main="Gamma distributed data",xlab="Percentile values",ylab="",font.lab=2)
lines(x,splinefit(y,xc=c(0,.4,.95))$F)
legend(.1,max(y)*.8, c("o Simulated claims","- Spline with exponential tail","Grid=(0,.4,.95)"),bty="n")
```

A.3.2: Table 5.1, 5.2 and 5.3

```
y=sort(danishfire)
y= sort(hurricane)
y=sort(danishfire)

spline=function(s,y,M,xc,N,u)
{{sum((s[1]*pmin(u,xc[N])+s[2:N]%*%M-s[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))-y)**2)}}

xc=c(0,.4,.95)
N=length(xc)
u=(1:length(y)-0.9)/length(y)
M=pmax(t(matrix(pmin(u,xc[N]),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)$par

m=1000
n=length(y)
emp=spline=1:m*0
for (i in 1:m)
{emp[i]=sum(sample(y, n, replace = TRUE, prob = NULL))
u=runif(n)
M=pmax(t(matrix(pmin(u,xc[N]),length(u),N-1))-xc[1:(N-1)],0)**3
y_spline=o[1]*pmin(u,xc[N])+o[2:N]%*%M-o[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))
spline[i]=sum(y_spline[,])}

floor(mean(emp))
floor(sort(emp)[.95*m])
floor(sort(emp)[.99*m])
floor(mean(spline))
floor(sort(spline)[.95*m])
floor(sort(spline)[.99*m])
```

A.3.3: Table 5.4 and 5.5

```
k=c(100,500,1000)
emp_mean=emp_95=emp_99=spline_mean=spline_95=spline_99=1:3*0

for (j in 1:3)
{y=sort(2*(runif(k[j])**(-1/2)-1))#Pareto distributed data
#y=sort(rgamma(k[j],2))#Gamma distributed data.


spline=function(s,y,M,xc,N,u)
{{sum((s[1]*pmin(u,xc[N])+s[2:N]%*%M-s[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))-y)**2)}}

xc=c(0,.4,.95)
N=length(xc)
u=(1:length(y)-0.9)/length(y)
M=pmax(t(matrix(pmin(u,xc[N]),length(y),N-1))-xc[1:(N-1)],0)**3
s=rep(1,N+1)
o=optim(s,spline,"BFGS",y,M,xc,N,u)$par

m=1000
n=length(y)
emp=spline=1:m*0
for (i in 1:m)
{emp[i]=sum(sample(y, n, replace = TRUE, prob = NULL))
u=runif(n)
M=pmax(t(matrix(pmin(u,xc[N]),length(u),N-1))-xc[1:(N-1)],0)**3
y_spline=o[1]*pmin(u,xc[N])+o[2:N]%*%M-o[N+1]*log((1-pmax(u,xc[N]))/(1-xc[N]))
spline[i]=sum(y_spline[,])}

emp_mean[j]=floor(mean(emp))
emp_95[j]=floor(sort(emp)[.95*m])
emp_99[j]=floor(sort(emp)[.99*m])
spline_mean[j]=floor(mean(spline))
spline_95[j]=floor(sort(spline)[.95*m])
spline_99[j]=floor(sort(spline)[.99*m])}
emp_mean
emp_95
emp_99
spline_mean
spline_95
spline_99
```

A.4: Empirical datasets used in thesis.

A.4.1: Norwegian male mortality data (datamalenorway2011)

```
datamalenorway2011=matrix(
  c(18.00000,    32.21473,  33839.00000,
    19.00000,    23.95801,  33983.00000,
    20.00000,    20.83189,   33873.00000,
    21.00000,    35.35176,  33163.00000,
    22.00000,    29.36461,  32555.00000,
    23.00000,    23.15188,  31160.00000,
    24.00000,    19.75175,  31502.00000,
    25.00000,    18.52033,  31179.00000,
    26.00000,    20.63622,  31267.00000,
    27.00000,    27.74778,  31894.00000,
    28.00000,    30.72248,  32545.00000,
    29.00000,    21.44784,  31964.00000,
    30.00000,    30.98709,  32895.00000,
    31.00000,    36.32570,  32667.00000,
    32.00000,    33.56770,  32813.00000,
    33.00000,    27.34750,  32098.00000,
    34.00000,    28.13395,  33216.00000,
    35.00000,    40.12481,  34207.00000,
    36.00000,    37.47400,  35826.00000,
    37.00000,    31.02500,  36500.00000,
    38.00000,    28.22756,  37687.00000,
    39.00000,    45.80374,  37917.00000,
    40.00000,    46.53416,  37649.00000,
    41.00000,    52.30770,  38575.00000,
    42.00000,    44.67083,  38410.00000,
    43.00000,    59.93984,  37439.00000,
    44.00000,    49.92997,  37683.00000,
    45.00000,    67.19802,  37085.00000,
    46.00000,    79.12038,  36579.00000,
    47.00000,    79.62371,  34755.00000,
    48.00000,    78.13337,  34254.00000,
    49.00000,    94.70632,  33560.00000,
    50.00000,    93.21718,  33197.00000,
    51.00000,    95.96637,  33368.00000,
    52.00000,   107.80385,  32817.00000,
    53.00000,   124.31214,  32122.00000,
    54.00000,   118.06561,  32276.00000,
    55.00000,   153.90349,  31557.00000,
    56.00000,   149.75554,  30738.00000,
    57.00000,   170.61627,  30321.00000,
    58.00000,   186.78000,  30000.00000,
    59.00000,   190.88008,  28773.00000,
    60.00000,   219.14761,  29111.00000,
    61.00000,   212.96876,  29110.00000,
    62.00000,   270.46356,  29427.00000,
    63.00000,   276.32405,  29484.00000,
    64.00000,   322.27713,  30510.00000,
    65.00000,   333.31994,  26846.00000,
    66.00000,   353.69553,  25658.00000,
    67.00000,   351.72755,  22305.00000,
    68.00000,   292.93064,  20083.00000,
    69.00000,   330.03878,  17104.00000,
```

```
 70.00000,   317.53375,  17588.00000,
 71.00000,   380.18922,  16507.00000,
 72.00000,   367.82953,  15498.00000,
 73.00000,   424.26899,  14289.00000,
 74.00000,   395.22223,  13453.00000,
 75.00000,   445.27596,  12196.00000,
 76.00000,   422.81054,  11812.00000,
 77.00000,   491.05917,  11445.00000,
 78.00000,   532.23262,  11566.00000,
 79.00000,   603.88957,  10758.00000,
 80.00000,   614.94455,  10449.00000,
 81.00000,   664.77808,   9426.00000,
 82.00000,   642.17209,   8781.00000,
 83.00000,   647.19316,   7854.00000,
 84.00000,   795.41584,   7442.00000,
 85.00000,   711.18956,   6622.00000,
 86.00000,   718.69611,   5891.00000,
 87.00000,   704.10907,   5023.00000,
 88.00000,   618.33110,   4266.00000,
 89.00000,   556.92441,   3387.00000,
 90.00000,   523.22858,   2979.00000,
 91.00000,   393.86332,   1932.00000,
 92.00000,   340.29888,   1562.00000,
 93.00000,   251.15584,   1091.00000,
 94.00000,   190.96376,    695.00000,
 95.00000,   146.33240,    496.00000,
 96.00000,   134.31002,    379.00000,
 97.00000,    58.06699,    222.00000,
 98.00000,    53.94511,    148.00000,
 99.00000,    27.48921,     81.00000,
100.00000,    18.29982,     51.00000,
101.00000,     8.48681,     24.00000,
102.00000,     4.78750,     12.00000,
103.00000,     1.63850,      7.00000,
104.00000,     3.10065,      6.00000,
105.00000,     1.65201,      3.00000),88,byrow=T)
```

A.4.2 Danish fire claims data (danishfire)

```
danishfire=c(1.68375,2.09370,1.73258,1.77975,4.61201,8.72527,7.89898,2.20805,1.48609,2.79617,7.32
064,3.36750,1.46413,1.72222,11.37482,2.48274,26.21464,2.00243,4.53001,
1.84175,3.80673,14.12208,5.42425,11.71303,1.51537,2.53859,2.04978,12.46559,1.73545,1.68375,3.323
85,1.82102,2.41581,1.46413,5.87590,1.79244,3.33821,1.58449,1.93132,2.04978,
1.57376,1.75695,4.93119,5.41728,1.53734,17.56955,1.69112,1.58161,1.46712,2.04978,1.92871,3.26315,
1.70425,2.07461,3.29429,1.83998,7.32064,7.32064,1.57622,4.85610,
1.54228,13.62079,1.75695,1.63982,2.03638,21.96193,1.79903,2.45242,2.66911,2.02415,1.72474,1.6438
6,1.96193,4.62665,1.46413,3.96325,5.56385,4.39239,3.64031,1.67789,
1.44949,263.25037,2.50073,1.82577,9.88287,2.14662,2.87026,6.31991,2.12346,2.92387,2.03638,1.7364
6,2.04978,1.90337,1.61054,1.71742,3.12944,2.05196,1.96752,2.10249,3.25915,
1.99854,5.41728,2.81984,3.69708,2.04539,2.67936,3.42622,1.58858,4.26287,1.75586,1.98701,1.66193,7
.61347,1.79488,6.67482,2.18594,3.85505,3.22108,2.12299,1.76029,2.04165,
3.75594,2.24211,1.72111,1.46413,4.71189,2.15382,3.87481,19.07028,1.88726,1.46413,1.53616,2.56007,
2.79971,2.01318,1.46413,6.32504,1.56730,2.19619,2.33529,2.11567,1.46954,
5.40264,5.31479,2.87262,2.78184,2.57682,3.10102,1.46413,2.04978,1.96193,1.46413,2.63543,1.46413,5
.85652,3.29429,2.52086,19.47291,1.83546,2.04978,2.78331,2.78184,1.75695,
2.84041,2.33089,1.75623,1.31455,2.40397,6.91563,1.49188,1.53240,1.87386,1.81520,8.25688,1.41020,2
.20663,34.14155,2.44331,3.59437,2.37221,3.66972,6.88845,1.80210,1.44168,
```

9.17431,2.09699,2.81455,1.68490,2.09961,2.62123,1.37615,1.66448,2.35911,4.13649,1.39440,1.80210,1
.81913,1.51376,3.08077,20.96986,4.19397,1.86164,2.49672,2.21378,5.24246,
1.32225,8.55177,1.96592,12.89515,1.78899,2.17683,8.77763,1.81520,1.68314,3.64351,1.61337,1.46789,
1.40560,3.12080,1.59818,3.80079,1.51009,1.57274,1.83005,2.62123,2.80648,
1.35780,1.61861,1.96592,3.02228,56.22543,4.98034,1.98716,1.96592,2.71129,1.90039,1.72346,1.57274,
1.70380,3.01966,1.83486,1.41940,4.39056,7.86370,1.57274,1.33661,1.70380,
2.63562,1.86472,1.45775,7.55832,2.03140,1.34993,1.47486,5.11140,5.24246,1.44168,3.93614,4.06291,1
.37615,1.68414,1.44062,1.36455,1.57274,2.18542,3.28135,1.36310,2.12320,
2.63434,2.09830,3.43435,1.47880,2.62123,3.00131,1.42973,10.22280,2.57940,1.75536,1.47340,1.95282,
1.77987,1.98686,1.93971,1.67590,1.57274,2.52490,14.67890,3.03408,1.97903,
5.93776,1.44168,2.62123,2.03801,3.75491,2.43775,1.70203,7.10987,2.01966,1.85049,1.41595,7.69332,3
.01442,2.35911,1.59232,1.38925,1.33297,1.42333,2.97757,1.38736,1.40498,
2.62123,4.27916,1.60550,2.01180,2.26737,1.46465,3.37407,1.49080,2.09699,1.57017,1.37615,3.14548,8
.45374,3.03328,2.06422,6.42202,1.89879,1.49017,50.06553,1.55963,1.96592,
1.80005,2.42464,1.31062,1.61206,4.22117,1.26040,1.40369,1.29727,1.92749,1.91546,1.20745,5.69858,1
.90250,1.18913,10.17802,1.60208,4.16171,2.36305,1.96504,1.94598,2.85375,
1.20244,10.82045,1.58876,2.09635,5.46968,1.89298,2.03329,1.60523,1.40309,1.19501,3.24614,1.26312,
2.97265,1.20495,1.88127,1.78359,3.92390,1.90250,1.66468,4.45283,1.32362,
1.60523,24.97027,2.73484,1.78359,1.81094,2.11415,1.39120,1.34602,1.50738,1.94049,2.56980,1.58077,
1.48751,1.52200,1.69084,1.78359,1.25887,1.57375,1.46730,2.71371,2.73603,
1.45660,2.13743,2.47909,2.37812,4.63734,1.18906,1.54578,2.34245,2.33769,1.39715,4.22117,1.63995,1
.21879,3.68609,2.49703,2.49703,5.23187,1.30797,1.26812,2.16528,4.64019,
2.97265,1.66468,1.18906,11.89061,2.86564,1.24437,1.72414,5.69560,1.35957,7.07491,1.40348,1.40242,
1.31986,4.32059,1.84304,1.41498,1.48633,4.94603,1.91320,20.04994,1.43651,
5.50773,3.21046,1.39933,5.00173,3.44828,3.55648,1.35757,2.13151,1.84189,1.42687,2.41760,1.25215,1
.89970,1.55918,1.24925,1.90324,1.60523,2.22966,1.21284,3.44828,1.24327,
1.67493,1.35315,1.78754,2.03686,1.97384,1.64090,2.90993,2.55096,1.80737,4.00258,1.42687,2.34311,1
.38882,4.17360,1.27854,3.55886,1.20015,1.55123,1.69962,65.70749,27.26259,
15.92628,1.27950,3.09844,2.94696,1.22473,1.66468,1.31960,1.51660,1.27323,1.66468,1.30797,5.46968,
1.24851,1.98683,1.42687,1.42687,2.16409,2.14031,1.30797,1.49838,1.68847,
1.83995,4.37574,1.78669,1.65594,1.18906,3.85076,4.16171,2.14031,2.14031,2.61593,4.16171,22.25823,
2.25922,4.28062,1.30625,5.58859,2.79429,1.44160,1.34965,1.56507,1.11235,
2.81016,1.78345,2.12870,4.34928,2.71079,1.13459,2.62369,2.39377,6.23471,1.37820,1.21691,1.16352,2
.45941,4.72747,1.55729,1.30701,3.14349,1.19711,1.16796,1.11235,1.77976,
1.64740,1.72369,5.56174,1.33482,1.44605,1.33482,1.59066,1.11235,3.03244,3.78643,2.22469,2.44716,1
0.01112,1.16355,1.11235,3.15286,3.72636,4.44939,2.28126,3.55951,4.44939,
1.20912,1.15684,2.22469,1.24027,1.15926,1.66852,3.33704,10.07230,2.14127,1.47497,1.11235,1.68966,
1.66852,1.61735,1.11235,2.16908,1.49388,1.40823,2.50278,2.25806,3.72636,
1.89370,1.70412,2.34285,1.94661,1.27920,1.51070,2.44716,3.67519,2.37824,1.77419,1.51513,2.23471,1
.81646,3.44828,4.67186,1.15684,1.22358,4.07893,1.19244,1.20409,2.55840,
1.61846,1.82202,1.61056,1.86096,7.99207,1.78532,4.89433,2.52058,1.39488,1.39043,1.29718,3.67075,3
.87653,3.58732,1.43795,1.17527,1.50167,1.50600,5.92547,12.63181,3.06118,
1.43604,2.27920,1.49177,6.91655,1.60423,1.66852,1.15721,1.32036,3.44828,1.97576,1.85651,2.00222,5
.30050,4.00445,1.12447,1.54649,2.55840,1.47917,1.25940,3.19800,3.11457,
1.31257,5.67297,13.34816,11.43159,4.25918,3.36683,1.11235,1.11235,1.44605,2.07311,1.22358,1.6129
0,1.31924,1.12903,2.33593,1.72414,11.12347,1.66852,1.55729,1.55729,1.93331,
1.72414,1.50167,2.20743,1.04712,1.42536,1.36126,9.31414,1.64489,2.61780,2.54111,1.57068,1.25654,1
.45864,1.04712,2.23141,2.19895,5.02618,2.40838,1.65218,1.15183,1.31968,
2.70157,2.54450,2.35602,1.25654,1.67539,1.36126,3.87435,1.76963,1.11832,2.20942,1.09843,1.15183,2
.96375,11.62304,14.29319,1.28796,1.46597,13.62304,1.12984,2.40314,5.44503,
1.04691,1.05916,2.09424,1.85550,1.57068,1.93717,2.20314,2.46073,1.42762,1.73508,4.08377,2.15707,1
.46597,1.16859,1.22513,1.15183,3.76963,1.09505,1.20419,2.61780,1.87120,
4.39791,4.60733,18.64648,1.04712,1.04712,1.04712,15.81152,1.98953,1.04712,1.25654,1.15183,1.0733
0,1.19442,1.06436,1.38429,1.13089,1.15183,1.79372,1.14136,1.63291,2.64398,
1.28690,4.45026,1.23665,1.95916,1.19105,1.31443,1.98953,1.64957,2.72251,1.13242,1.78010,1.60733,1
.20419,1.07330,1.09948,1.40628,1.96545,3.36319,1.57068,1.25654,2.76244,
1.06649,1.37795,1.30025,1.31937,3.40314,1.36126,1.17801,4.29319,4.80838,1.09948,2.21990,2.61780,1
.43450,1.04712,2.51309,1.62146,19.16230,1.57068,2.30366,1.84555,1.04712,

1.98953,2.82723,1.25654,3.97906,1.36126,1.63351,18.84817,1.12565,5.30558,1.72775,1.16534,1.15183,
1.25654,3.56859,7.23560,1.57068,7.64398,2.12200,2.93194,1.28796,1.06801,
1.27561,1.62304,2.67016,1.25654,1.10305,1.20419,1.15183,1.88482,7.53927,1.09948,1.57068,2.67016,1
.15183,3.87435,5.02618,1.78010,4.76440,1.15183,1.50000,1.25100,1.03000,
1.05000,1.90000,1.10000,1.88175,1.00700,1.63000,1.02500,1.00727,3.50000,2.90000,2.46314,4.62500,1
.03000,1.40000,1.07708,1.32000,2.60000,1.08000,1.51700,1.07000,2.25682,
1.78841,1.27890,4.60907,3.18300,1.20000,2.74300,3.00000,1.00894,1.00408,1.85000,1.52099,1.25009,1
.00000,1.95600,1.05000,2.77900,22.13757,1.44500,1.00510,1.00000,1.00000,
16.30000,1.11000,1.47500,1.04987,1.75354,6.14335,1.23000,1.93900,46.50000,1.24147,3.40000,1.0000
0,6.20000,1.96624,2.17000,1.29600,1.55500,1.15000,1.33000,4.12429,1.39424,
4.76123,4.86885,4.00000,1.60000,7.08500,1.76505,6.30665,1.25000,1.20000,1.22641,6.56300,4.17000,2
.68000,1.07500,1.93296,1.37500,1.25000,3.38000,1.38010,2.28300,2.35000,
3.15000,3.35000,4.00000,10.50000,2.89000,5.50000,2.80000,1.26000,1.10000,2.12676,2.20000,1.07000,
9.20000,1.50000,1.65000,12.22500,1.40000,14.23900,1.08500,1.05000,2.65000,
2.12300,5.20000,1.82500,1.10000,1.00000,2.06500,1.27740,1.19500,1.20000,1.10000,1.40000,1.58500,3
.26504,1.20000,2.43300,1.60000,2.00000,4.76000,1.16500,1.37200,1.20000,
1.65000,1.70000,1.30000,1.10000,1.10000,1.24101,1.00500,3.28200,1.05000,57.41064,1.08800,1.11800,
2.67000,1.07490,2.14100,5.85000,14.30000,1.97100,1.70000,6.16700,1.05000,
1.15606,3.58215,13.50000,1.20150,1.65000,1.37760,3.80000,1.32550,4.05000,6.70000,1.16000,1.50000,
1.06000,1.49129,1.98181,1.50000,3.21436,4.10000,1.00000,10.70000,19.40000,
1.60560,1.13200,2.25500,2.02000,1.23100,4.10000,7.23000,1.85000,1.00000,2.56000,1.55300,1.70000,1
.04547,1.05400,1.00000,8.71000,1.50000,1.50000,1.26106,1.05500,1.28200,
1.30000,1.30000,1.42208,1.00000,1.20000,1.85000,1.18000,1.04300,1.00000,1.60000,5.60000,1.80000,2
.50000,1.75000,1.45000,1.46095,1.15718,1.49576,5.20733,3.68370,1.44648,
1.03866,1.25362,1.06075,1.39826,1.39826,2.41080,1.44185,1.63934,1.05063,1.75506,4.48409,1.67792,1
.06075,1.00000,3.08293,1.27772,1.20733,1.92864,1.00289,1.08968,1.75813,
1.44648,1.59113,1.10897,1.02311,2.00096,1.31431,2.06847,3.08582,6.79846,1.63934,1.63934,1.22951,1
.73578,2.82342,2.50723,1.86674,4.43587,1.92864,1.06075,1.74147,3.08582,
12.05400,1.15718,2.41080,1.55834,1.35005,3.18226,1.90066,2.31437,3.70781,3.20636,1.06075,1.44648,
1.15718,5.20733,3.66442,1.65923,2.50723,3.19190,16.44166,1.08293,2.41080,
3.66925,1.18129,29.02604,4.91803,1.14851,1.54291,1.56702,1.25362,1.54291,1.83221,1.66345,1.27772,
2.44359,2.89296,3.13404,2.60366,3.78978,1.90260,3.77044,1.16683,1.18901,
1.32112,1.73578,12.53616,1.32112,1.15718,5.78592,1.83221,1.38898,1.20540,1.10897,1.72936,1.63934,
1.08765,1.54291,1.43202,2.16972,5.40019,5.78592,1.35005,6.07522,18.32208,
1.15718,1.59113,1.92864,1.44648,1.67792,1.15718,1.01100,2.84474,1.73578,1.15043,2.50723,2.89296,2
.09740,2.08650,1.44648,2.15029,1.12209,1.65863,1.69720,1.59113,5.78592,
1.05593,1.44648,2.89296,1.15718,1.12054,1.15718,1.20338,5.59306,1.43298,1.26432,1.39923,1.14770,1
.23823,1.75988,1.54291,2.12150,1.54291,1.25362,1.92864,1.33076,1.92864,
1.10897,2.02507,8.67888,2.21794,1.30183,1.12390,1.72758,1.37416,1.00696,8.10029,1.00289,1.17165,1
0.27001,1.27140,2.89296,3.34523,1.06075,2.89296,1.12150,1.69720,1.71649,
1.10241,17.06847,2.60366,1.25362,3.56798,1.49470,1.63934,1.72228,2.35101,1.68756,1.47059,2.31437,
1.25362,1.14443,1.20540,2.60270,2.94179,2.18129,1.44648,4.82160,3.83799,
1.35005,1.54291,1.15718,1.92864,2.24204,1.26326,1.01254,1.92864,1.13790,1.03761,1.25362,2.18756,4
.76730,1.54291,1.06075,2.75217,3.18226,1.44648,1.92864,1.59113,3.08582,
4.09836,1.88042,2.41080,1.01254,1.92864,1.06075,1.63934,1.12825,1.44648,1.44648,2.89296,1.38862,1
.63934,2.02218,1.70516,1.05111,17.74349,1.15718,3.85728,1.70516,1.35005,
5.49662,1.02319,1.30427,1.06679,1.80891,3.24675,1.41744,2.49072,1.01113,2.97124,3.00464,1.42022,2
.23377,1.39796,2.31911,2.85250,1.99443,1.74212,1.64657,2.97310,1.11317,
1.48423,1.11317,1.14935,3.18460,2.22635,3.89610,1.43785,2.36827,1.02134,1.02041,2.73655,23.28386,
1.39147,1.39147,2.04082,1.02041,2.15306,1.06679,1.39147,1.45918,4.99072,
1.05751,4.12801,1.07699,1.50278,1.02134,4.50000,3.10761,2.25417,1.75788,1.39889,1.25232,4.49814,2
.41187,2.10575,3.18831,3.68460,1.57699,1.65121,2.52319,1.66976,1.90445,
1.85529,1.62338,1.85529,2.31911,1.43785,3.03803,1.25696,1.24490,3.89610,1.11317,1.28015,6.58627,1
.08534,1.60668,1.29870,1.76252,1.11317,1.25046,1.23562,1.11317,4.82375,
2.42579,3.14750,1.61874,1.52134,1.85529,7.14286,1.21707,1.11317,1.57699,1.62338,1.76252,1.39147,1
.15955,11.13173,1.11317,1.85529,1.66976,12.52319,2.04082,5.75139,5.56586,
5.75139,1.11317,4.17440,8.81262,3.78015,32.46753,2.08720,3.29314,3.33952,1.39054,1.62338,4.45269,
1.05380,1.39147,1.39147,1.66976,1.69017,1.31725,4.40631,1.39147,2.13358,

1.20594,1.36364,2.39796,3.24675,1.06957,2.87570,1.02041,1.11317,1.81633,1.35622,1.48423,2.78293,1
.61410,29.03711,18.55288,2.50464,3.06122,1.16419,1.11317,1.36364,1.07607,
1.51206,1.84787,16.88312,1.39147,1.90167,1.44898,1.76809,1.02041,1.57978,1.20594,1.51670,1.02041,
1.85529,1.20594,1.02041,1.22449,1.00464,7.79221,1.34972,27.82931,4.17440,
5.89981,1.00649,1.50742,1.20594,1.11317,12.05937,2.31911,3.33952,1.15955,2.37291,1.09647,1.10482,
1.25232,3.65028,1.39147,1.29870,6.30798,9.46197,2.87570,1.68367,3.48145,
7.60668,1.39147,3.89610,1.12523,11.59555,3.07699,2.11503,1.85529,1.75696,1.48423,3.06122,1.57699,
1.73469,1.76252,1.28942,1.46846,5.19481,1.94805,4.11874,4.96289,5.10204,
1.34508,1.07514,7.10111,2.07050,2.41187,1.23748,2.04082,1.29870,2.89425,1.06679,1.39147,1.53061,1
.04824,1.17811,1.34508,2.04082,1.66976,1.61846,1.02751,16.41526,1.50843,
1.68589,1.33097,2.30701,1.92014,3.77107,1.15350,5.09406,2.95652,1.79858,1.55280,1.33097,1.88731,1
.77462,1.33097,6.01597,3.26176,1.16681,2.52085,1.29370,18.42413,1.06477,
1.95563,2.04082,7.98580,3.49689,6.51109,1.25998,1.85359,1.55280,1.26708,2.12955,2.70630,1.55280,1
.33097,1.46761,2.30701,1.90772,1.24224,1.21118,1.99645,38.15439,2.12955,
2.48447,1.69033,1.76575,1.53860,1.68589,1.24224,1.33097,1.98225,1.86335,1.88554,5.21029,1.15350,1
.78882,5.76752,1.33097,6.83230,1.36202,3.54925,1.37533,2.30701,2.66193,
7.54215,5.68146,9.22804,1.08075,1.61224,1.24224,1.35670,1.50843,2.30701,1.37533,1.55280,27.33807,
1.23780,1.19255,1.27773,1.33097,7.54215,2.22360,1.15350,1.78793,1.43744,
6.14020,1.61668,2.09406,11.80124,2.97249,1.95209,2.48802,3.99290,2.22272,25.28838,2.44011,3.7267
1,3.75244,1.64153,1.24224,1.24224,4.43656,1.06477,1.19787,2.39574,10.20408,
1.77462,2.32032,2.30701,1.08784,1.14286,1.15350,1.86690,2.48447,2.57320,1.12156,5.98935,2.50932,1
.32831,1.04525,1.38776,1.15350,1.28660,1.04703,4.61402,20.45253,2.21828,
5.32387,1.21650,7.09849,1.33097,1.77462,3.01242,47.01952,1.10027,2.30967,1.14552,1.37622,1.14286,
2.89618,1.77462,1.35404,24.57853,2.26176,4.43656,1.34871,15.88287,1.02041,
1.26264,2.66193,1.05146,1.95209,1.35759,1.64153,7.54215,2.21828,5.50133,2.21118,1.47737,1.64153,1
.02041,3.28305,25.95386,1.06477,4.08163,2.28838,1.10027,2.83851,3.19432,
1.59716,1.77462,1.72138,2.33363,4.49423,8.87311,2.67613,1.14729,1.33097,1.50843,1.24224,1.10914,6
.21118,1.73026,1.15350,1.05590,1.87933,1.40195,1.41970,2.71517,4.08163,
1.75244,1.08873,3.54925,1.28305,2.17391,2.83940,3.37178,1.33097,4.61402,10.82520,5.76752,1.10914,
31.05590,3.10559,1.33097,1.68589,3.99290,1.84081,1.28281,5.37680,1.36156,
1.05843,1.00339,2.96359,1.27096,1.90517,4.23370,1.94750,2.91109,24.55546,1.22777,1.21931,3.26842,
4.82642,1.65961,1.21423,1.01609,1.52329,2.68417,1.56224,1.12108,1.01439,
42.09145,1.69348,1.64014,1.94750,4.65707,1.05843,1.11600,1.52413,1.39204,1.22777,14.39458,1.1304
0,1.74428,1.94750,1.18544,2.56562,1.14818,2.17612,1.33108,20.86367,1.89416,
2.11685,1.65114,1.05843,9.22947,1.14310,5.92718,1.19814,1.77477,1.87553,5.66469,1.10076,4.14903,1
.28366,1.05843,2.20152,1.28959,1.35478,4.41660,1.27011,14.39458,2.68671,
1.69348,1.62405,3.38696,4.06435,1.30144,1.27011,1.55377,1.47671,1.18967,1.01609,10.13717,2.66300,
1.04572,3.81033,1.27011,1.69348,1.48180,1.52413,2.96359,1.18120,1.56562,
3.04826,3.33785,1.45724,1.01609,4.30144,2.03218,4.02202,1.12193,1.87976,2.20152,1.27011,1.73412,1
.16850,6.77392,1.17697,4.50042,2.21338,1.01609,1.86283,2.20152,1.21931,
5.71550,6.43522,5.50381,2.54022,5.08044,5.08044,2.03218,4.09145,1.90178,4.82642,2.13209,1.56986,1
.25318,1.27011,3.09907,1.91025,10.29551,2.54022,2.48772,2.28620,1.40559,
1.10246,1.01609,3.85267,12.80186,1.60881,1.52413,1.97290,1.27011,1.78493,16.08806,1.43946,152.41
321,1.43946,3.13040,6.28704,2.77731,1.03302,1.01609,14.01355,1.27011,1.21507,
8.36749,1.05843,1.11770,3.81033,1.10076,1.52075,1.48772,3.38696,4.65707,2.27773,1.04149,1.17697,1
.52413,1.28874,1.27011,3.77392,1.19814,2.35224,1.06689,1.06689,9.39881,
1.27011,5.77053,1.04996,1.53683,1.08383,1.87976,1.27858,1.35478,3.97968,5.24979,1.56562,12.70110,
4.57240,2.92125,1.77815,1.52413,11.68501,2.11685,1.73328,2.54022,1.17612,
6.01185,32.38781,10.58425,1.64268,3.38696,1.01609,4.86876,1.52837,1.10076,1.07959,1.69348,18.628
28,1.73158,3.81033,1.10076,1.83743,1.01609,4.65707,1.31499,1.52413,2.28620,
1.01609,1.88569,1.31160,1.22777,1.10076,3.38696,1.63421,1.01355,1.72650,3.81033,2.45639,2.68671,1
.00339,1.01524,1.86283,1.08383,2.11685,4.14903,1.01609,1.01609,1.17697,
1.87706,1.26815,1.35644,7.42574,1.13449,1.32921,1.28713,1.56518,1.07261,2.31023,1.09406,2.61716,1
.99010,1.98762,1.15594,3.37871,1.21700,1.95132,4.12541,1.40264,1.02640,
1.70792,1.23762,3.30033,1.27888,2.98102,4.07591,3.71287,1.23762,1.03135,1.45462,1.68977,2.36799,3
.46535,1.23762,1.06436,1.42327,1.48515,2.47525,1.73680,10.99835,1.08498,
1.15512,1.25825,1.89769,1.00660,1.10891,1.03135,1.32013,1.90264,8.25083,1.09488,2.88779,2.30281,1
.73597,3.21782,1.08498,1.48515,2.49917,4.86799,1.10561,12.37624,1.74670,

```
3.58911,1.02145,3.46535,4.53795,5.19802,2.67574,1.27558,1.65017,3.30033,1.32013,1.48515,2.53300,3
.85561,1.20875,1.29043,2.80528,1.26568,5.61221,1.44389,2.10396,2.47525,
1.81518,1.49505,2.04290,1.07096,1.23762,2.06271,1.01733,1.65017,15.28465,2.29373,1.07261,4.53795,
1.23762,10.18482,1.66419,2.31023,6.37294,1.41089,1.11469,1.87046,2.06271,
1.06848,1.67079,1.31848,1.11551,2.06271,1.07261,4.29043,2.98267,1.02145,8.08581,3.17657,1.65429,1
.26568,1.08168,1.08416,2.80528,2.16337,1.27888,1.32096,1.33911,1.48350,
1.82261,6.20050,13.20132,4.31436,2.40759,1.36469,1.00578,2.14521,1.90182,1.46122,20.82673,5.9405
9,2.72277,2.76238,1.02723,1.14191,1.64604,1.77063,4.12541,1.98020,1.32013,
1.00660,8.25083,1.11386,1.65017,1.26650,1.55363,4.08416,1.44884,14.85149,1.59653,3.87789,2.02145,
1.65017,1.01073,1.48515,1.27888,1.02310,4.70297,3.13531,1.86469,2.03795,
2.14521,1.15512,1.05611,144.65759,28.63036,19.26568,5.52805,1.54290,4.62046,1.07261,3.71287,1.01
403,1.11386,2.33993,1.93894,1.23762,3.05281,1.40924,2.55363,5.77558,1.13449,
3.40759,1.07261,1.16749,1.07261,1.27063,1.47277,1.03630,1.65017,1.67822,2.64026,1.60149,17.73927,
4.37294,1.36139,1.18399,2.97030,1.02310,1.13036,3.01155,1.40264,2.32261,
1.11551,1.69142,1.23762,1.11469,1.40264,4.86799,1.07261,4.12541)
```

A.4.3 Belgian fire claims data (belgianfire)

```
belgianfire=c(6.525,7.652,13.978,23.548,14.709,20.615,19.613,27.088,3.459,14.448,4.898,46.598,16.802,
0.052,19.480,46.613,9.572,4.507,25.085,75.203,19.930,66.165,34.913,39.017,46.640,4.222,5.653,11.639,
3.479,45.708,15.825,3.209,96.378,13.386,35.706,27.448,2.494,35.361,1.485,9.130,29.438,0.329,12.982,
34.482,2.366,5.257,61.281,3.165,3.995,52.349,1.693,67.511,1.547,2.133,7.078,15.654,11.657,13.342,
3.773,5.285)
```

A.4.4 US Hurricane claims data (hurricane)

```
hurricane=c(6.766,7.123,10.562,14.474,15.351,16.983,18.383,19.030,25.304,29.112,30.146,33.727,
40.596,41.409,47.905,49.397,52.600,59.917,63.123,77.809,102.942,103.217,123.680,140.136,192.013,
198.446,227.338,329.511,361.200,421.680,513.586,545.778,750.389,863.881,1638.000)
```